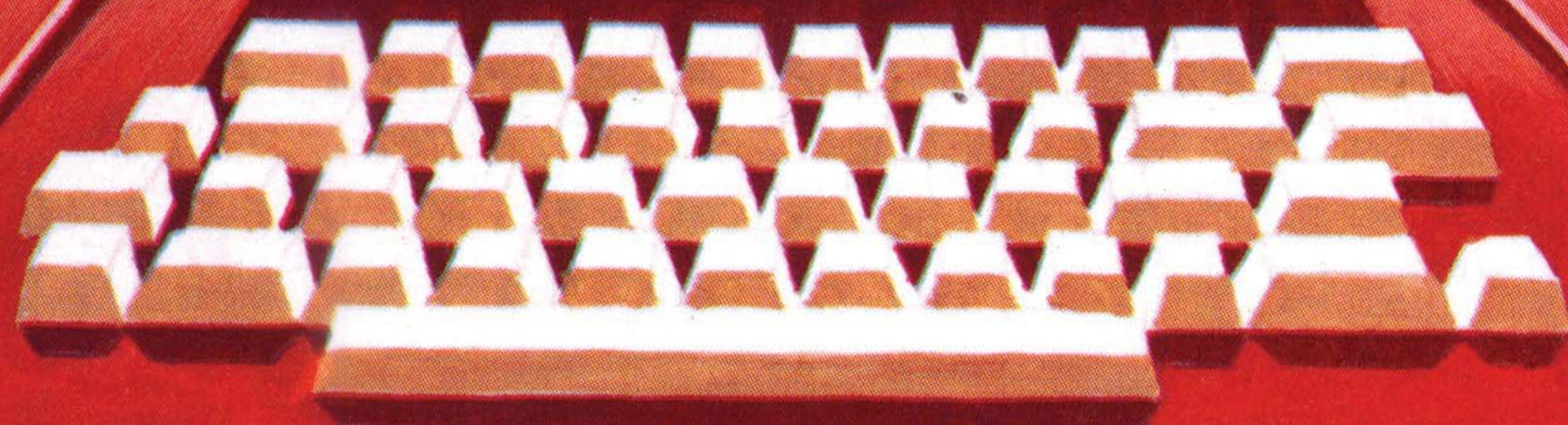
UNA PUBBLICAZIONE
DEL GRUPPO EDITORIALE JACKSON

ANNO 3 N. 9

Bit

MICROPROCESSORS-HARDWARE - SOFTWARE
HOME & PERSONAL COMPUTERS

L. 2000 ★

Spedizione in abb. postale Gruppo IV/70

ANTEPRIMA APPLE III

NELLA RUBRICA
COMPUTER PRATICO
PET ED APPLE

MERCATO COMPUTER

MUSICA
ELETTRONICA
E MICROCOMPUTER

IL DIGITIZER
CORSO SUL PASCAL
DATA BASE PERSONALE
GIOCO DI SIMULAZIONE

NEC

il miglior floppy disk controller

Ci sono due modi per valutare la semplicità, la potenza e la flessibilità del nostro floppy disk controller: usare il componente UPD 765 o la piastra BP-2190 della NEC.

Il nuovo componente UPD 765 sviluppato dalla NEC è già diventato uno standard dell'industria ed ha già un secondo fornitore.

E' semplice da introdurre nel sistema perchè è compatibile con il floppy tipo IBM a singola o doppia densità e con il mini-floppy 5 1/4" e con lo standard da 8".

E' potente perchè esegue 15 comandi complessi incluse varie subroutines del software di governo del disco e perchè controlla fino a 4 comandi su due lati.

E' flessibile nel programmare il Vostro sistema di controllo grazie a comandi come il Multi-Sector Read/Write, Track Formatting, Multiple Drive Seeks, usando la via del DMA o dell'Interrupt e interfacciandosi con i microprocessori più noti (UPD8080AF, UPD8085A UPD780/Z80™).

La piastra BP-2190 aggiunge altre prestazioni a quelle sopra descritte, perchè include il componente UPD 765 e una RAM da 16K bytes (espandibile a 48K bytes) con logica di priorità e rinfresco. Il trasferimento da disco a RAM è sotto il controllo del DMA, impegnando il microprocessore ospite al minimo.

La piastra può essere direttamente connessa al sistema Multibus™ e può gestire varie densità e dimensioni.

Chiedere informazioni dettagliate a:

Ufficio: **NEC Electronics Italiana s.r.l.**
Via Cardano, 3 - 20124 Milano - Tel. 02/630368-632646
Distributore: SILVERSTAR
Via dei Gracchi, 20 - 20146 Milano - Tel. 02/4996 (12 linee)

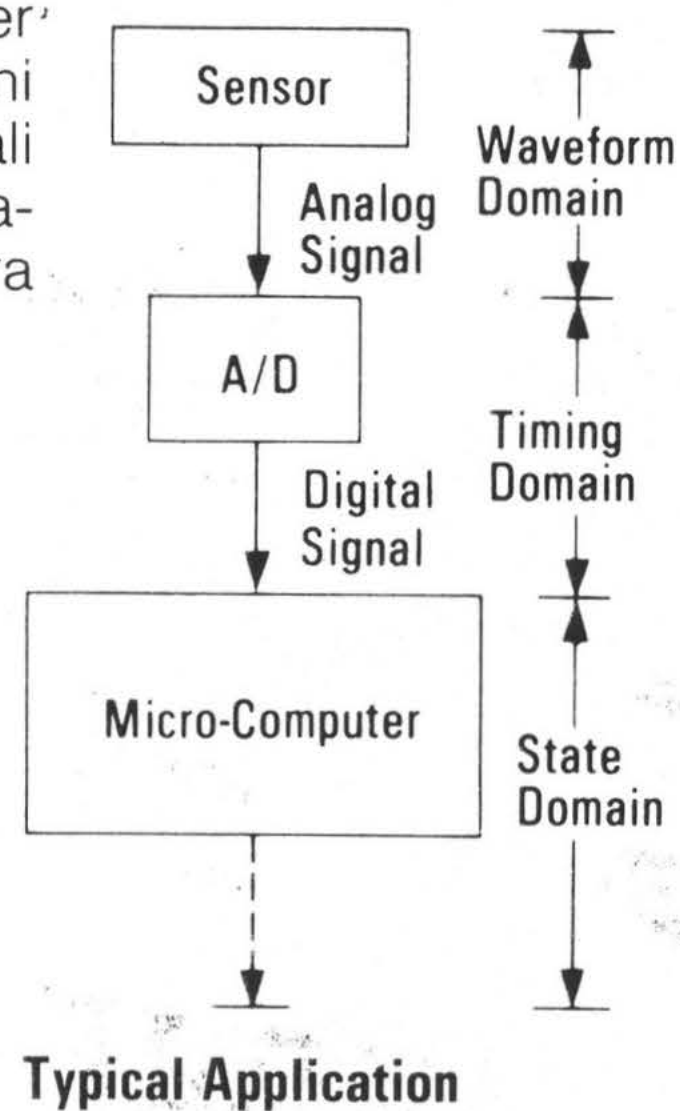
ANALISI DIGITALE/ANALOGICA
IL SISTEMA DI MISURA DEGLI ANNI 80

Presentiamo il nuovo sistema di analisi logica Paratronics PI540

Quanto più intelligente sarà il Vs. prodotto negli anni 80, tanto più difficile sarà il provarlo. Fino ad ora avete usato analizzatori di stati e di tempi per risolvere problemi digitali sincroni ed asincroni. Ma, oltre ai segnali logici, vi sono anche segnali analogici e pertanto Vi occorre tuttora un registratore di forme d'onda.

Ed ecco la terza generazione
La Paratronics, studiando le necessità degli anni 80, ha realizzato il PI540 in modo da combinare in un unico strumento compatto le funzioni di rappresentazione degli stati, dei tempi e delle forme d'onda. Il nostro nuovo Sistema di Analisi Logica Vi fornisce 32 canali di stato, 8 canali addizionali di tempo (o stato) ed un canale analogico - il numero ed il tipo di funzioni di rappresentazione a Voi necessarie per progettare e provare i prodotti attuali a bus. Ma non bastano le capacità di rappresentazione solamente: la grande novità è che Vi consentiamo di correlare qualsiasi o tutte queste possibilità d'analisi per scovare i problemi «inter-dominiali» che incontrerete. Per esempio, nell'analisi di controllori di processo a μP, le capacità combinate del PI540 Vi consentono di seguire sequenze inizianti con un ingresso analogico, proseguenti con la conversione in digitale e terminanti nel risultato finale dello svolgimento del programma.

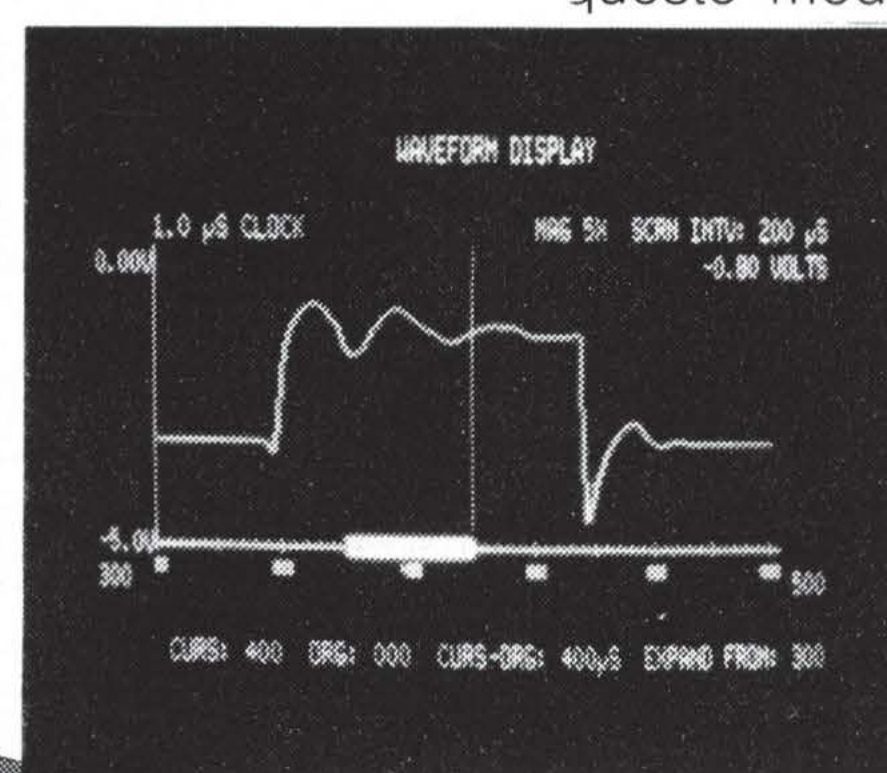
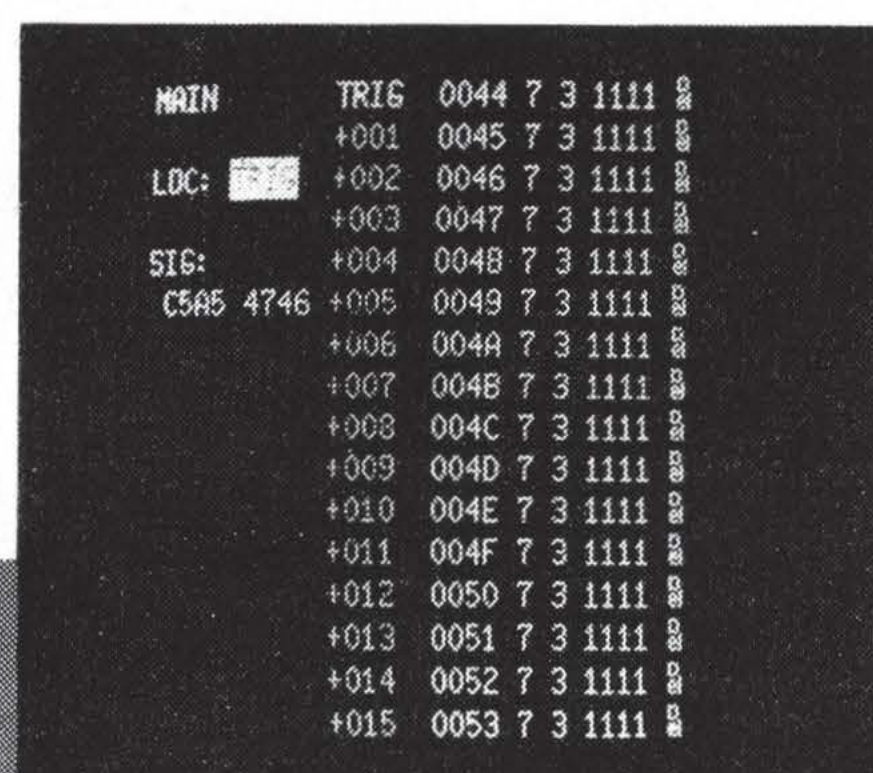
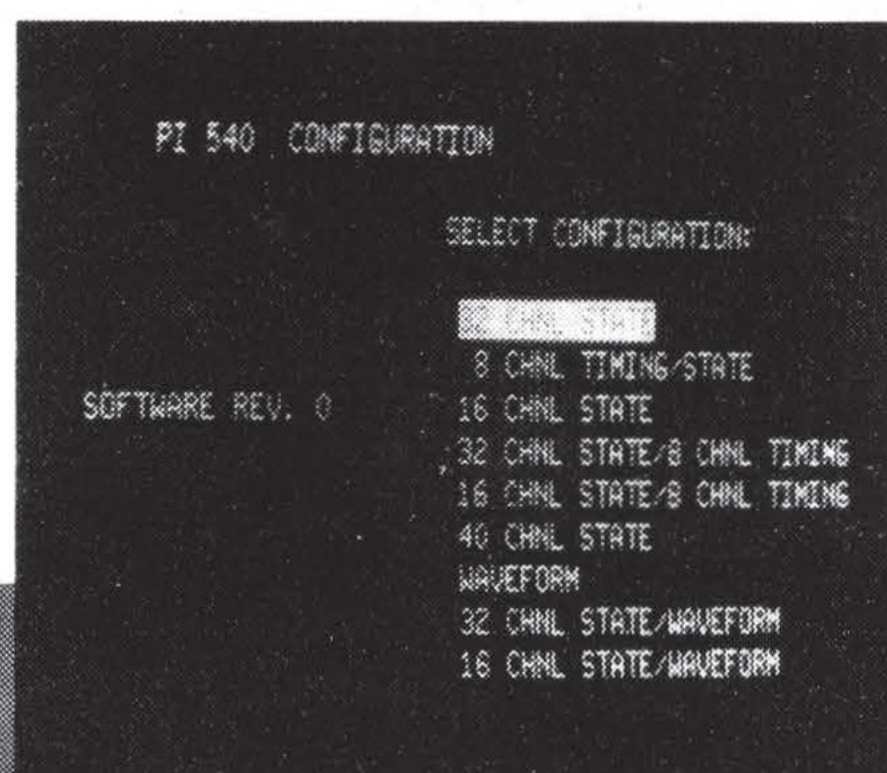
Sensor
Analog Signal
A/D
Digital Signal
Micro-Computer
Waveform Domain
Timing Domain
State Domain

Typical Application

Lo strumento di analisi più completo esistente
Le prestazioni normali del PI540, comprendono sonde a soglia variabile, 16 livelli di trigger, signature analysis, processi di correlazione, memorie di paragone e formati di rappresentazione esadecimale, ottale, binario, decimale, ASCII, in tempo e in forma d'onda.

Un cenno circa l'unità base Sistema 5000
Il PI540 è contenuto in una unità base Paratronics Sistema 5000 che comprende un ampio CRT da 23 cm (9 pollici), una tastiera piatta, con funzioni anche protettive e con pulsanti in rilievo ad azione positiva per una sicura impostazione dei dati ed un sistema a microcalcolatore di uso generale. Le singole funzioni di analisi, esclusive del PI540, sono modulari, inseribili quindi a schede standard nello spazio previsto all'interno del sistema 5000. In questo modo il PI540 può venire fornito con le capacità di analisi che servono ora e completabile poi per successive esigenze.
Se volete garantirVi di mantenere il funzionamento intelligente delle Vostre macchine intelligenti negli anni 80, vorrete sapere tutto sul PI540. Per informazioni complete o per una dimostrazione contattare la ditta VIANELLO S.p.A. a Milano o Roma.

PI 540 CONFIGURATION
SELECT CONFIGURATION:
SOFTWARE REV. 0
32 CHNL STATE
8 CHNL TIMING/STATE
16 CHNL STATE
32 CHNL STATE/8 CHNL TIMING
16 CHNL STATE/8 CHNL TIMING
40 CHNL STATE
WAVEFORM
32 CHNL STATE/WAVEFORM
16 CHNL STATE/WAVEFORM

WAVEFORM DISPLAY

SYSTEM 5000

PARATRONICS INC.

Vianello
AGENTE ESCLUSIVO PER L'ITALIA

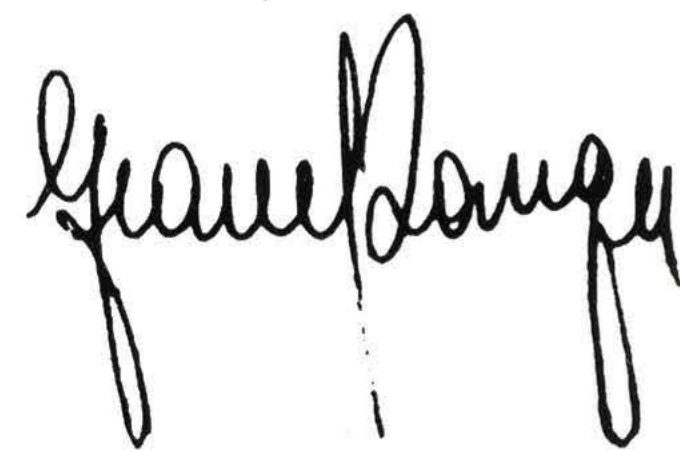
Sede: 20121 MILANO - Via T. da Cazzaniga 9/6 - Tel. (02) 3452071 (5 linee)
Filiale: 00185 ROMA - Via S. Croce in Gerusalemme 97 - Tel. 7576941/250

Fin troppo umani

Troppi anni fa esisteva un periodo felice in cui il problema principale che ci assillava era di decidere chi fra noi, tutti ovviamente menti eccelse, fosse provvisto dell'intelligenza più elevata. Sorvolando allegramente sulla definizione di "intelligenza", avevamo deciso che il metro giusto era il gioco degli scacchi; e allora via con pomeriggi interi passati in interminabili tornei, da cui il vincitore usciva con un soddisfatto eh eh, il perdente con sordi brontolii di "non ero in forma" ed entrambi con il cervello completamente intronato.

Le prime avvisaglie della sconfitta umana passarono quasi inosservate: un amico, discreto scacchista e già vincitore di tornei, mi aveva raccontato affranto di essere stato invitato da altri amici in un centro di calcolo, di essere stato seduto a viva forza (diceva lui) davanti ad un monitor e di aver perso con ignominia una partita a scacchi contro sì e no una tonnellata di computer; ricordo di aver pensato che una partita non faceva primavera, che insomma doveva aver fatto qualche cretinata imperdonabile, capita nelle migliori famiglie.

Giorni fa ho rivisto lo stesso amico; pallido, emaciato, profonde occhiaie e sguardo perso nel nulla. Era chiuso e taciturno come un'ostrica, ma dopo un paio di bicchieri è venuta fuori l'intera, tremenda storia: la Sciagura gli si era presentata in casa sotto forma di un grazioso pacchettino per il suo compleanno. Apertolo avidamente, si era trovato davanti ad uno scatolino dall'aria innocente e un po' dimessa, con una scacchiera serigrafata sulla faccia superiore, una tastiera da quattro soldi ed un display anch'esso piuttosto economico. Era uno scacchista elettronico.

Sprezzante, il mio amico aveva impostato il massimo livello di difficoltà (sembrava solo una stupida macchinetta) e aveva incominciato una partita; alla quattro del mattino successivo la partita si era conclusa con la sua dignitosa, ma definitiva sconfitta. Da allora erano trascorse un paio di settimane di partite a scacchi, interrotte solo per mangiare qualche panino, dormire qualche ora e lavorare controvoglia; alla moglie che proponeva il divorzio già per la terza volta aveva risposto "Sì, cara, mmm, G5-E6". Il resto del mondo per lui aveva assunto la consistenza diafana e dubbia di un fantasma.

Il colpo peggiore gli era stato poi inferto in uno dei rari momenti di lucidità quando, da buon elettronico, aveva aperto la scatoletta infernale per vedere cosa c'era dentro; aveva gli occhi sbarrati nel dirmelo: una CPU Z80, una ROM, un paio di RAM (1k in tutto) e due chips di periferia per la tastiera e il display. Tutto quì.

Dove l'esser stato battuto da un CDC da una tonnellata non era stato poi tanto umiliante, il continuare a vincere e a perdere in pari misura con meno di un etto di Z80 aveva spezzato duramente il suo orgoglio di Uomo.

Superato l'attimo di panico (i computers ci stanno mangiando) ed il successivo senso di colpa (io sono uno degli strumenti di cui quelle dannate fette di silicio si stanno servendo per conquistare il mondo), ci ho ripensato.

È la vecchia storia della ruota del vasaio, la cui invenzione ha reso obsoleti quei vasai che sapevano fare i vasi rotondi anche a mano. Ma non ha portato all'estinzione né l'umanità, né i vasai. Continuo ad essere convinto che l'Uomo sia difficilmente sostituibile nelle sue manifestazioni fondamentali, anche se come parte in causa non posso essere certo della mia obiettività.

Ma se un giorno i fatti mi dovessero dare torto, se l'Uomo quel giorno potesse venir sostituito interamente da un computer, vorrà dire che quel computer sarà abbastanza umano da poter essere chiamato Uomo lui stesso.

P.S. *Ogni riferimento a persone e fatti è reale.*

A parte questo, stiamo esaminando le risposte inviate tramite il questionario, che, dopo la valanga delle prime settimane, continuano ad arrivare ora a sgoccioli. Per non far torto a nessuno includeremo anche i ritardatari (che siano i lettori più accaniti?!) nel sorteggio degli abbonamenti gratuiti.

Comunque già in questo numero abbiamo realizzato parte dei vostri suggerimenti.

A coloro che ci hanno scritto per collaborare una raccomandazione: stiamo prendendo contatti con tutti, ma le cose saranno più semplici se a questo punto ci manderete una breve descrizione del vostro lavoro. Ogni articolo pubblicato sarà ovviamente ricompensato.

STARPLEX™

Sottosistema video
Ampio schermo da 12 pollici • Caratteri di ottima definizione - matrice di punti 7x9 • 24 linee per 80 caratteri • Visualizzazione in verde su fondo nero • Intensità variabile dello schermo • Inclinazione dello schermo di 10° per una lettura più comoda • Ampio controllo dello schermo: scrolling, lampeggiamento, blank, inversione

Sottosistema processore
La CPU basata sulla scheda BLC/SBC 80/204 • Floppy disc controller/formatter • CRT Controller • Memoria utente da 64K byte • Quattro slot per future espansione

Tasti funzione del sistema
8 tasti per il controllo dell'esecuzione del programma • I tasti non usati sono programmabili da parte dell'utente

Tasti di editing
5 tasti controllo cursore • 13 tasti speciali di editing

Pulsante Reset e caricamento bootstrap
Bootstrap residente con micro-diagnostico per il check completo del sistema in fase di inizializzazione, con successivo switch verso la memoria utente

Tastiera ASCII
58 tasti alfanumerici

PROM programmer (opzionale)
Programmazione delle PROM bipolari e delle EPROM 2708, 2716

Tasti di selezione programmi
10 tasti di selezione programmi • 2 tasti di selezione specializzabili dall'utente

Sottosistema stampante
Matrice di punti 5x7 • Stampa termica silenziosa • 80 caratteri per linea • 50 caratteri al secondo • Caratteri full e half size • Avanzamento della carta

Sottosistema disco
Sistema a due floppy con 256K byte per driver • Formato IBM soft sectored • Espandibile a quattro driver (1.024 KB)

Le caratteristiche fondamentali e nuove dello STARPLEX:

- Facilità di utilizzo, basato su una operatività da tastiera a funzioni complesse, potendo con un solo tasto, porre in esecuzione richieste dell'operatore, che prima richiedevano una più lunga procedura.
- Software potente, costituito da Fortran IV, Basic, sistema operativo, Macro Assembler, Editor, Debugger.
- Supporto completo alla linea di prodotti NS della serie BLC 8010 e dei prodotti 8080 A, 8070, 8048...
- Modularità, sia nella configurazione fisica, che nel software.
- Affidabilità elevata; viene fornita una garanzia di 90 giorni.
- Basso costo, in quanto le ampie prestazioni dello STARPLEX costano molto meno delle ormai obsolete prestazioni di analoghi sistemi della precedente generazione di sistemi di sviluppo.
- Schede di espansione BLC 80, schede utente.
- Completa integrazione Hardware-Software.
- Opzione: ISE (In System Emulator), PROM programmer.

National Semiconductor

Per un sistema globale

• National Semiconductor, Milano (02) 3452046/7/8/9

AGENTE: **Repco srl**, Milano (02) 4985274-4985932-4985494, Roma (06) 8107788
DISTRIBUTORI: **Adelsy spa**, Milano (02) 4524651, Genova (010) 589674, Udine (0432) 26996, Padova (049) 45600-45778, Torino (011) 539141, Roma (06) 594559
• **E.D.L. spa**, Napoli (081) 632335, Bari (080) 365461 • **Esco Italiana**, Milano (02) 6072441
• **Fanton**, Padova (049) 654487 • **Intelco**, Bologna (051) 726186, Firenze (055) 608107
• **Inter-Rep spa**, Torino (011) 752075 • **Intesi**, Milano - S. Donato Milanese (02) 51741, Roma (06) 2275130-223372, Torino (011) 613963 • **Side srl**, Ancona - Osimo Scalo (071) 79307
DISTRIBUTORE SISTEMI DI MEMORIE: **ESE srl**, Milano (02) 6073626

Per favore speditemi ulteriori informazioni su STARPLEX
National Semiconductor
Via Solferino, 19 - 20121 Milano

Nome: ______

Posizione: ______

Società: ______

Via: ______ Città: ______

Bit

Anno 3 - N° 8/9 - Giugno/Agosto 1980

In copertina:
Musica e Computer
Grafica di Renato NISI.

Sommario

INDICE INSERZIONISTI

AVVISO AI LETTORI

Le continuazioni degli articoli "Il microcomputer nelle applicazioni gestionali" e "Trasmissione seriale" saranno pubblicate nel numero di Settembre.

DIRETTORE RESPONSABILE
Giampietro Zanga

COORDINATORE TECNICO
Marcello Montedoro

CAPO REDATTORE
Pietro Dell'Orco

COORDINAMENTO PERSONAL COMPUTER
Gloriano Rossi

SEGRETARIA DI REDAZIONE
Cecilia De Serio

GRAFICA E IMPAGINAZIONE
Gianfranco de Rienzo

DIFFUSIONE E ABBONAMENTI
Gabriella Napoli, Silvia Decari
Marco Benedetti

DIREZIONE, REDAZIONE
P.le Massari, 22 - 20125 Milano
Telefoni 68.03.68 - 68.00.54

AMMINISTRAZIONE
Via Vincenzo Monti, 15 - 20123 Milano

PUBBLICITA': Concessionario per l'Italia e l'Estero Reina & C. S.n.c.
Via Ricasoli, 2 - 20121 Milano
Tel. (02) 803101-866192-8050977
864.066 - Telex: 320419 BRUS I

Concessionario pubblicità per USA e Canada:
International Media Marketing
16704 Marquardt Avenue
P.O Box 1217
Cerritos CA 90701
(213) 926-9552

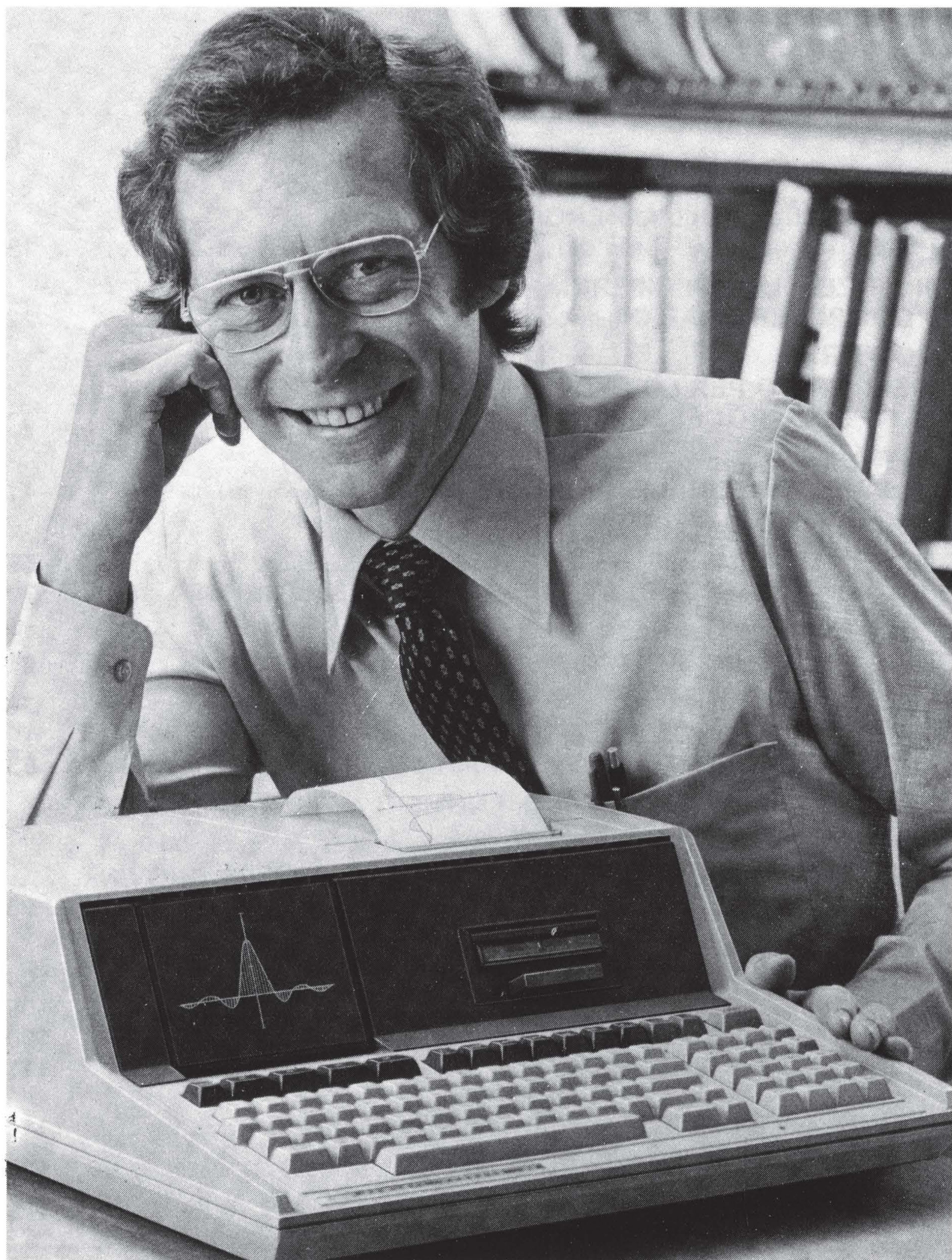
STAMPA: Litografia del Sole srl
Buccinasco

Concessionario esclusivo per la diffusione in Italia e all'Estero:
SODIP - Via Zuretti, 25 - 20125 Milano

Spedizione in abbonamento Postale Gruppo III/70

Prezzo della rivista L. 2.000

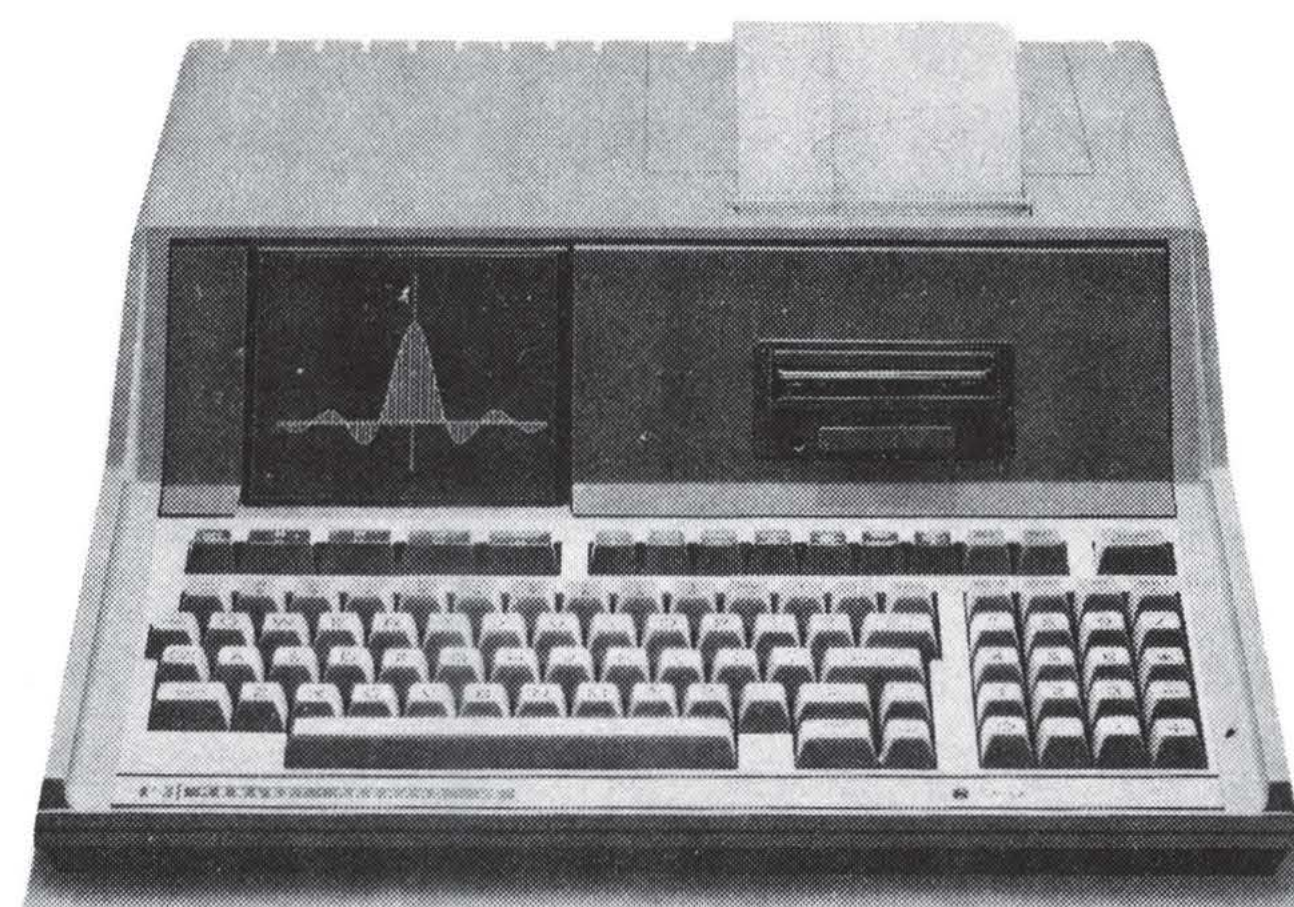
Numero arretrato L. 3.000
Abbonamento annuo L. 12.500
per l'Estero L. 17.500
I versamenti vanno indirizzati a: Jackson Italiana Editrice S.r.l.
Via V. Monti, 15 - 20123 Milano
mediante emissione di assegno bancario, cartolina vaglia o utilizzando il c/c Postale numero 11666203
Per i cambi di indirizzo, indicare, oltre naturalmente al nuovo, anche l'indirizzo precedente, ed allegare alla comunicazione l'importo di L. 500, anche in francobolli.

Il personal computer Hewlett-Packard. Lavora come un grande computer. Ma solo per te.

Il personal computer HP-85 ti mette a disposizione una piena potenza di elaborazione a livello professionale ovunque ti serva: sulla tua scrivania, in laboratorio, perfino a casa tua.

Con tutti i vantaggi di un Sistema compatto e facile da usare.

Da Hewlett-Packard non potevi aspettarti di meno.

Un singolo "corpo" non più grande di una macchina per scrivere comprende un video grafico, una stampante ad alta velocità, una unità a nastro magnetico, il sistema operativo e la tastiera.

E una volta nella sua valigetta, l'HP-85 è anche facile da trasportare.

Completo, compatto, tuo.

Non è solo simpatico e semplice da usare: è anche incredibilmente versatile, con ampie possibilità di periferiche e di programmazione.

HP-85, il personal computer Hewlett-Packard. Studiato per risolvere i problemi di calcolo, dei professionisti della finanza, dell'ingegneria e della scienza.

Rivenditori Autorizzati: Bimac, Bologna - tel. (051) 261956; Frongia Mario, Cagliari - tel. (070) 650756; M.B.M. s.a.s. dell'Ing. Musumeci & C., Catania - tel. (095) 445988; Clup s.c.r.l., Milano - tel. (02) 230668; Homic, Milano - tel. (02) 437058; Svemar, Napoli - tel. (081) 312050; Rag. Enrico Capovilla & C. s.a.s., Padova - tel. (049) 28998; Univers Elettronica s.r.l., Roma - tel. (06) 779092; Meccanografica Torinese, Torino - tel. (011) 238803.

hp HEWLETT PACKARD

Newsletter

La Apple apre in Europa uno stabilimento e un centro di distribuzione

La Apple Computer Inc. di Cupertino (California - USA) avvierà, il 1° settembre prossimo la produzione in Europa dei propri personal computers ed aprirà un importante centro di distribuzione ed assistenza.
Lo stabilimento produttivo sarà aperto a Cork, in Irlanda, coprirà una superficie di 4.000 metri quadrati ed inizierà l'assemblaggio del personal computers Apple II.
Nello stesso tempo diverrà operativo a Zeist, in Olanda, un centro di distribuzione e di assistenza, allo scopo di servire la clientela europea.
La presenza in Europa con uno stabilimento di produzione rappresenta un passo importante nel quadro del programma espansionistico della società, che prevede altri nuovi impianti in Irlanda e nel Texas (USA) e sei centri regionali di assistenza, fra i quali è compreso quello olandese.
Mike Scott, presidente della Apple, è certo che l'essere presenti in Europa contribuirà a meglio servire il sempre crescente numero di clienti europei.
L'impianto irlandese sorge su un'area di 18 acri: nei piani della società l'area produttiva - attualmente di 4.000 mq. - sarà ampliata a più di 33.500 metri quadrati nei prossimi cinque anni.
Per il primo anno di attività verranno impiegate 65 persone ma il totale dovrebbe aumentare sino a raggiungere le 1.400 unità circa nel 1985.
La Apple è uno dei due maggiori fornitori di personal computers nel mondo. Il suo ben conosciuto Apple II ed il nuovo personal computer Apple III sono attualmente distribuiti in Europa attaverso una rete di 13 distributori nei maggiori Paesi.

Pascal e LISP per il PET

Sono disponibili anche in Italia i linguaggi Pascal e LISP per i computers PET.
Il linguaggio di programmazione Pascal, richiede come sistema minimo una unità centrale CBM 3032 a 32 kbytes di memoria, ed un doppio drive floppy 3040.
Il Pascal è un linguaggio completamente compilato, quindi il sottosistema è strutturato in due parti: editor e compilatore.
Se entrambi sono presenti nella memoria del sistema, la memoria utente disponibile è di circa 5800 bytes per il programma sorgente (più lo spazio di memoria per il programma compilato).
Se il sottosistema Pascal viene fatto lavorare in "overlay" automaticamente di volta in volta, carica la parte di sistema interessata (editor o compilatore) lasciando a disposizione dell'utente circa 28900 bytes di memoria tutti a disposizione del programma sorgente.
Il compilatore, oltre ad eseguire la compilazione, la ricerca degli errori e la visualizzazione degli stessi, crea su disco il file di deposito chiamato "file oggetto" che può essere mandato in esecuzione in qualsiasi momento rimanendo all'interno dell'editor Pascal.
Oltre a tutte le funzioni presenti nell'editor Basic e nel Basic Plus, l'editor Pascal permette l'accesso al disco per la creazione ed il richiamo dei files sorgente ed oggetto Pascal.
Da programma Pascal è possibile in qualsiasi momento il richiamo e l'esecuzione di routines in linguaggio macchina e l'accesso alla memoria del video che il programma Pascal riconosce come matrice VDU con due indici ROW e COL.
Il LISP è stato definito l'intelligenza artificiale, poichè più di qualsiasi altro linguaggio si avvicina al modo di pensare dell'uomo.
La sua struttura tratta le istruzioni ed i dati allo stesso modo, permettendo in pratica di generare un programma attraverso un altro programma.
È il linguaggio di programmazione più utilizzato a livello scientifico e didattico/universitario, ma la sua semplicità ne permette l'utilizzo ancha a coloro che non sanno ancora programmare.

Newsletter

Il linguaggio PL/65 per l'AIM 65

PL/65, un linguaggio ad alto livello per realizzazione di sistemi, è adesso disponibile per il microcomputer Rockwell AIM 65 PL/65 disegnato per incrementare la produttività del programmatore e per aumentare l'affidabilità del programma. Le istruzioni per il controllo, come le esecuzioni condizionali (IF-THEN-ELSE), i cicli condizionali (FOR-TO-BY), affiancati da una possibilità di gestione dei blocchi semplificata, aiutano le tecniche di disegno del programma strutturato.
Il compilatore PL/65 genera un codice sorgente di linguaggio assembler R6500. Inoltre permette che le istruzioni del linguaggio assembler vengano incorporate direttamente in porzioni di programma PL/65, dove le richieste di ottimizzazione della temporizzazione o del codice sono critiche. Il risultato è un linguaggio per realizzazione di sistemi che ha il potere e la flessibilità del linguaggio assembler e il potenziale strutturale di un linguaggio ad alto livello.
Il compilatore PL/65 dell'AIM 65 è contenuto in due ROM da 4 kbyte che si inseriscono direttamente nel modulo principale dell'AIM 65.
È annunciata inoltre una pubblicazione periodica per tutti gli utilizzatori di questi microcomputer i quali potranno abbonarsi presso la Electronic Devices Division Rockwell International Corporation. La pubblicazione, chiamata Interactive propone la collaborazione di tutti i lettori dei quali pubblicherà articoli, domande e risposte; saranno inoltre promosse le attività dei gruppi di utilizzatori e pubblicati articoli scritti, specificatamente su nuove applicazioni.

Una nuova iniziativa della Tandy

Sul mercato della elaborazione personale la Tandy Corporation di Fort Worth è stata la società che ha saputo, fino ad oggi, fare più di tutti e ciò grazie alla rete di distribuzione della divisione Radio Shack forte di circa 7600 punti di vendita. Complessivamente si calcola che questa società abbia venduto più di 200.000 sistemi TRS-80, pari a circa la metà dei personal computer complessivamante venduti. A partire dallo scorso giugno la Tandy Corp., spronata dagli ottimi successi collezionati con la serie di calcolatori TRS-80, ha deciso di estendere le sue attività inserendosi in un business dalle prospettive molto interessanti. Si tratta del Videotex. In Europa, questo servizio ha già raggiunto una certa notorietà mentre negli Stati Uniti esso è ancora ai primi passi per se negli ultimi mesi si è assistito ad un notevole risveglio. La Tandy ha captato la possibilità di ripetere il grosso successo mietuto con i personal conputers e così a partire dai primi di giugno ha lanciato una massiccia campagna promozionale a favore di questo nuovo prodotto al quale, forse un pò per scaramanzia e un pò per non tradire la linea del passato, ha dato il nome di TRS-80 Videotex. Di che si tratti è presto detto. Di un sistema in grado di trasformare il televisore domestico (come display) ed il telefono (come mezzo di trasmissione) in un sistema per l'interrogazione di data-bases contenenti informazioni sul tempo, gli sport, le quotazioni borsistiche, i programmi cinematografici, gli orari ferroviari ed aerei, nonchè eventuali altri servizi che nel tempo verranno sviluppati da apposite organizzazioni.
Il TRS-80 Videotex consiste in una "scatola nera" che connette il televisore ed il telefono, in una piccola memoria ed in una tastiera. È a partire da quest'ultima che l'utente richiede agli archivi elettronizzati le informazioni di suo interesse, visualizzate sullo schermo televisivo. Tale prodotto costa 400 dollari, un prezzo che esperti della Diebold Inc. giudicano un pochino eccessivo e comunque destinato a flettere se si vuole veramente creare un mercato di grosso consumo. Ancora una volta la forza della Tandy per imporre al pubblico questo prodotto sta nella sua vasta, quasi capillare rete di commercializzazione. La società, parallelamente alle azioni propagandistiche e promozionali di questi giorni, assume già le ordinazioni. Tra un anno

o giù di lì il TRS-80 Videotex sarà disponibile su tutti i suoi 7600 "banchi di vendita". In pari tempo la Tandy progetta di mettere prossimamente in commercio, ad un prezzo vicino ai 200 dollari, un decodificatore per permettere ai possessori dei personal computer (come pure altri apparecchi del genere) di adattare questi a funzioni di accesso ai database.

La gamma della Ohio Scientific

Questa società dispone di una completa ed avanzata gamma di prodotti. La Ohio viene considerata il solo costruttore in grado di coprire la totalità del mercato del micro.
Fino al 1979 i sistemi più venduti furono il CE-4P, il C2-8P e il C3. I prezzi variavano dai 975 dollari del primo sistema ai circa 5.000 dollari del C3 versione OEM (32K di memoria, due dischetti e la possibilità di collegarsi con quattro unità Winchester con una capacità complessiva di oltre 280 Mbytes).
Con il 1980 il C2-8P ha cambiato nome e viene ora presentato come il solo personal computer capace di venire collegato ad un sistema telefonico automatico di risposta e di chiamata così come ad un sistema di allarme. Il suo prezzo si aggira intorno ai 2600 dollari.

Il Komma lo si trova da Computer Shop

Fra i personal computers in vendita nel nostro Paese si trova il Komma, reperibile presso Computer Shop, sviluppato dalla società inglese Sirton Products. Da questo fornitore il Komma viene oggi importato ma tra la Sirton e la Detron s.r.l. (di cui Computer Shop è un marchio) esiste un accordo perchè a breve alcune fasi del montaggio vengano svolte in Italia. Di questa macchina, che si rivolge per lo più alla clientela OEM industriale e gestionale nonchè all'utenza finale con esigenze scientifiche, sono in vendita tre versioni, tutte dotate di memorie RAM da 64 Kbytes e con ampia gamma di periferiche. Varie e numerose risultano, quindi, le configurazioni.

La DEC come la Computerland?

Fra le società che sono entrate nell'arena dei personal computer durante il 1979 figura la Digital Equipment. I sistemi offerti da questa casa sono in genere basati sui noti LSI 11 e risultano orientati alla gestione. Secondo alcuni osservatori la DEC sarebbe intenzionata a costituire una rete di marketing analoga a quella della computerland, l'organizzazione considerata più in vista sul mercato della microinformatica. Nata nel 1976, la Computerland si è dedicata per un paio d'anni alla vendita di prodotti per hobbisti. Dal gennaio '79 essa ha esteso la sua attività verso altri segmenti. Oggi conta più di 80 negozi, per la maggior parte situati negli Stati Uniti. La tecnica di vendita è quella del franchising: i servizi comuni sono forniti dal centro. Sebbene gli acquisti siano centralizzati esiste una notevole interdipendenza tra la sede e la succursale. In cambio dei servizi la sede si rifà applicando una percentuale sul giro d'affari.
Una dependance della Computerland esiste a Parigi ed una seconda a Londra.

Con gli IAPX la Intel micromainframe

Un mainframe su una piastrina di silicio.
Quasi ci siamo. Entro l'anno la Intel, nell'ambito di un vigoroso programma di rinnovamento tecnologico e di marketing, presenterà il suo prima sistema da

32-bits. Trattasi di una macchina articolata su tre chip, decisamente innovativa e molto orientata al software. Si chiama iAPX 432 o micromainframe. Nella struttura ricorda il Sistema 38 dell'IBM. Non sarà questa la sola novità autunno-inverno dell'azienda di Santa Clara. Per l'inizio del 1981 essa programma l'introduzione di tre nuovi microprocessori: gli iAPX 186 e 188 a 16-bits e l'iAPX 286 a 32-bits nonchè il cambio di denominazione all'8086 e all'8080 in iAPX 86 e 88.

La formula P.A.S. della Zelco

La Zelco ha formulato e messo in pratica da qualche tempo una politica di assistenza che soddisfa le esigenze fondamentali del cliente pur lasciandogli la possibilità di usufruire dei vantaggi derivanti dall'acquisto di apparecchiature professionali altamenta affidabili, ossia bassi costi di manutenzione e bassi tempi morti per le riparazioni.
Per chiarire meglio i concetti sopra esposti si ricordi che il contratto di manutenzione viene spesso richiesto onde coprire le incognite derivanti dall'acquisto di una apparecchiatura nuova o che bene non si conosce piuttosto che per l'espletamento della normale manutenzione preventiva, in realtà quasi inesistente (almeno per la parte elettronica).
La Zelco ha creato la formula P.A.S. (Pay At Service o Paga Al Momento del Servizio) che consiste in un versamento minimo alla stipulazione del contratto, con la rimanenza della cifra da suddividersi e versarsi quando e se si verificano degli interventi. ciò inoltre scoraggia l'utente a chiedere interventi dovuti a non guasti o difetti degli apparati, bensì a false manovre o inadeguata preparazione dell'operatore. (È noto infatti che coloro che ricevono un calcolatore od un nuovo apparato cominciano spesso a "lavorarci " senza aver letto, come è invece opportuno, i relativi manuali).
La formula P.A.S. viene praticata dalla Zelco solo su materiale Zilog e terminali Soroc e tiene conto dell'alta affidabilità di essi.
Oltre ai contratti di tipo P.A.S. sono ovviamente ottenibili dalla Zelco i normali contratti di manutenzione e gli interventi singoli.

Uno stabilimento Philips per microcomputer

Ad Amburgo, in Germania, la Philips Gloeilampenfabrieken, ha messo in attività un impianto completamente attrezzato per la produzione dei microcomputer su singolo chip 8048 e 8021 per i quali c'è una domanda superiore all'offerta. La famiglia 8048 realizzata dalla casa olandese risulta totalmente compatibile con quella prodotta dalla Intel Corp. Nello stesso stabilimento verranno successivamente realizzati altri microcomputer.

25.000 le stampanti HISI prodotte nel 79

La Honeywell Information System Italia ha partecipato alla fiera di Hannover 1980 con un proprio stand dedicato alle stampanti seriali a matrice, la linea di prodotti che recentemente si è affiancata a quella, degli elaboratori general purpose medio piccoli, tradizionale della società.
Nate per la connessione con i sistemi Honeywell, questa stampanti, progettate nel Centro di Ricerca e Progettazione di Pregnana Milanese e prodotte negli stabilimenti di Caluso, hanno saputo conquistarsi in pochi anni (le prime stampanti sono uscite nel 1975) una posizione di assoluto rilievo sui mercati OEM di tutto il mondo, grazie soprattutto alla linearità del progetto e all'avanzatezza delle soluzioni tecnologiche impiegate. Circa 25.000 stampanti sono state prodotte lo scorso anno (e per

l'80% esportate) dagli stabilimenti di Caluso che, con la loro capacità di produzione odierna di quasi duecento stampanti al giorno, si pongono come la maggiore struttura produttrice del settore in Europa.
Ad Hannover sono stati presentati i due modelli più recenti: l'S10 e l'R28. La S10 è il più piccolo modello della serie e risulta particolarmente indicata nei collegamenti con mini e professional personal computer come stampante d'uscita, nonchè nei collegamenti con qualunque tipo di sistemi come stampante hard copy connessa a terminali video. Le sue caratteristiche sono: 80 colonne, una velocità di 80 caratteri al secondo, stampa bidirezionali e interfaccia seriale (EIA RS 232C) o parallela.
La R28 costituisce invece il modello top della serie ed è stata progettata e costruita per offrire agli utilizzatori OEM una stampante a matrice ad alte prestazioni ed elevate funzionalità. Le caratteristiche di questa stampante sono infatti tali da sofddisfare le applicazioni più esigenti.

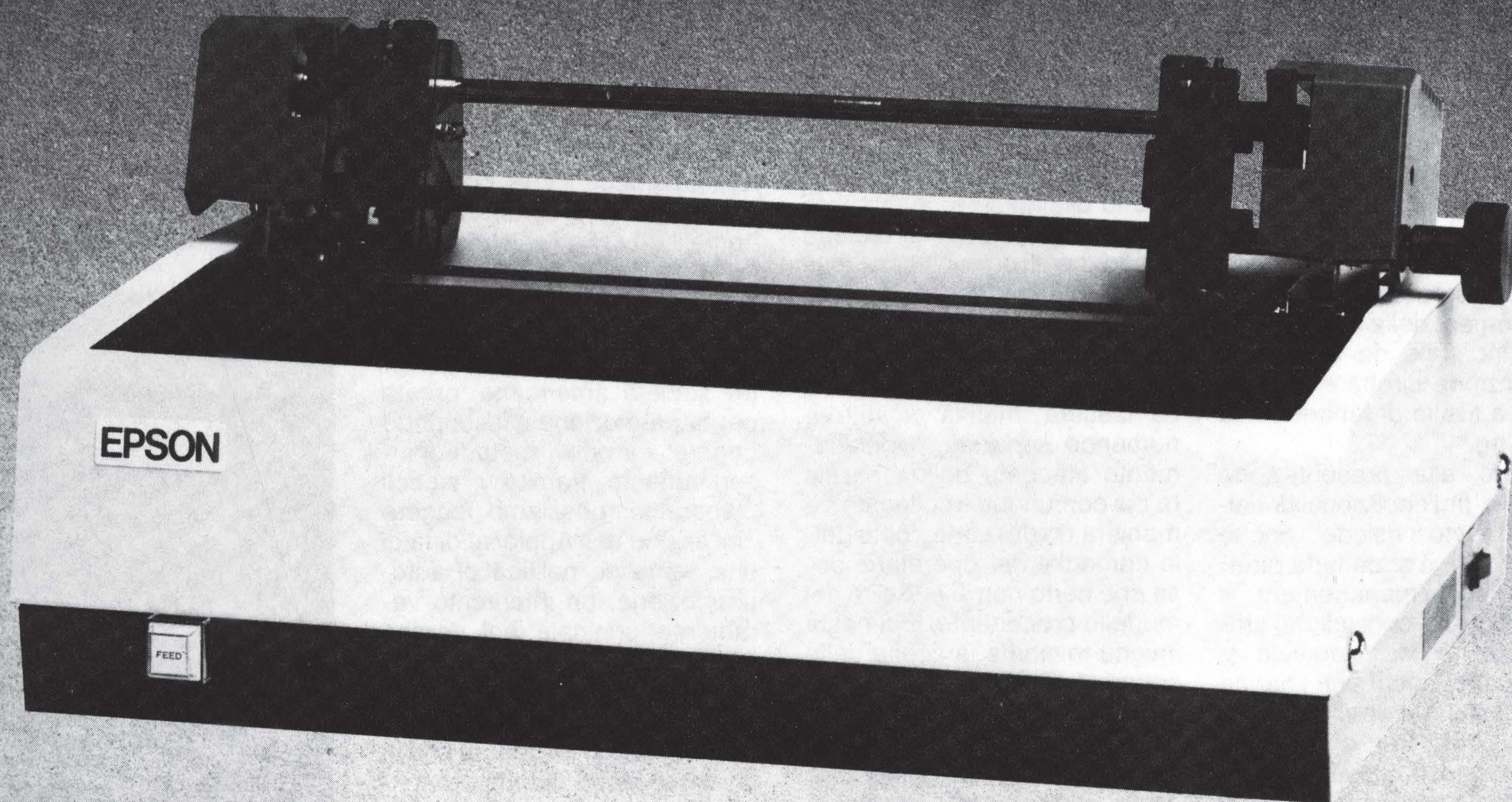
Il comando a microprocessore entra nelle motrici

L'impiego di azionamento di nuovo tipo nei treni locali (azionamenti a variatore in corrente continua o a correre alternativamente con motori asincroni), ha determinato la realizzazione di comandi elettronici più complessi e di maggiore dimensione; che possono tuttavia essere standardizzati e semplificati con l'impiego dei microcomputer. Il primo prototipo con comando a variatore in c.c. costruito dalla Siemens AG per treni metropolitani, è attualmente in prova presso l'UESTRA di Hannover. Questo comando è destinato a sostituire quelli elettronici del sistema Simatic, impiegati da oltre un decennio nelle automotrici dei treni locali.

È RIO il migliore

Secondo uno studio comparativo condotto dalla PA Computers and Telecommunications per la Central Computer Agency del governo inglese sui sistemi operativi per microcomputer, "il sistema operativo RIO della Zilog, realizzato per il microelaboratore Z80, sembra complessivamente quello capace di fornire le prestazioni più complete tra quelli oggi disponibili".
I sistemi operativi considerati, oltre al RIO, sono stati: CP/M, ISIS II e RMX/80 (Intel), MDOS (Motorola), TX990 (T.I.), DOS della Data General, TR-11 (DEC), CBM (Pet), DOS della Altair, Flex (6800), Famos (Z80), DOS della North Star e Trout.
La valutazione è stata fatta basandosi su una serie di criteri rigurdanti tre diversi aspetti del software di sistema:
— mezzi di sviluppo dei programmi (e linguaggi disponibili)
— inizializzazione e configurazione automatica del sistema (generazione)
— operazione e caratteristiche "run-time".

Software shop

La massiccia richiesta di software, venutasi a creare insieme con il grande successo dei "personal computers", ha dato lo spunto alla bits & bytes", 20124 Milano - Viale Monte Grappa, 4, per creare una serie di programmi in linguaggio basic per il TRS - 80 della Radio Shack americana.
Questa azienda copre con i suoi programmi ogni campo, da quello gestionale a quello finanziario, da quello scientifico a quello ricreativo.
Trovando una valida risposta nel mercato italiano, la "bits & bytes", seppure di nuova costruzione, ha potuto arricchire il suo bagaglio di esperienze, offrendo un'alta professionalità e ricorrenza alle reali esigenze di coloro che hanno già avuto l'opportunità di conoscerla.
È interessante notare, che per i programmi su nastro l'incisione su nastri magnetici viene effettuata sempre più di una volta, avendo cura di utilizzare esclusivamente in lato del nastro, e ciò al fine di evitare possibili sovrapposizioni di segnali con le note disastrose conseguenze.

di **Marcello Montedoro**

A brevissimo tempo dal suo debutto in America (in occasione della National Computer Conference), è avvenuta la presentazione in Italia dell'ultima novità della Apple Computer, Inc.: l'Apple III. Che questa non sia la norma è un fatto risaputo, per cui l'eccezione si carica di significati ulteriori se collegata ad altre notizie ufficiali di fonte Apple: l'apertura per settembre di uno stabilimento di produzione in Irlanda e la creazione di un centro di assistenza e distribuzione in Olanda, che sarà operativo alla medesima data.
A questo punto l'indicazione che emerge è molto chiara: il mercato europeo è ritenuto dal management della Apple Computer, Inc. tale da richiedere un'attenzione diretta e più efficiente, a livello di fabbrica e di marketing.
Tornando alla presentazione dell'Apple III l'eccezionalità dell'avvenimento risiede anche nel fatto che è stata fatta direttamente dal management: il presidente del consiglio di amministrazione Mr. Markkula e, gioia di quei pochi che l'hanno visto (la stampa specializzata), Steven Jobs. Per chi ancora non lo sapesse, Steven Jobs è, insieme a Stephen Wozniak, il fondatore dell'Apple: nel lontano (si fa per dire) 1976 Jobs e Wozniak erano la società Apple, con sede,nel garage di Jobs!; nel '79 il fatturato della Apple, Inc. è stato di 75 milioni di dollari, e si prevede che, entro l''80, i dipendenti arriveranno ad 800 unità: siamo già nella leggenda.

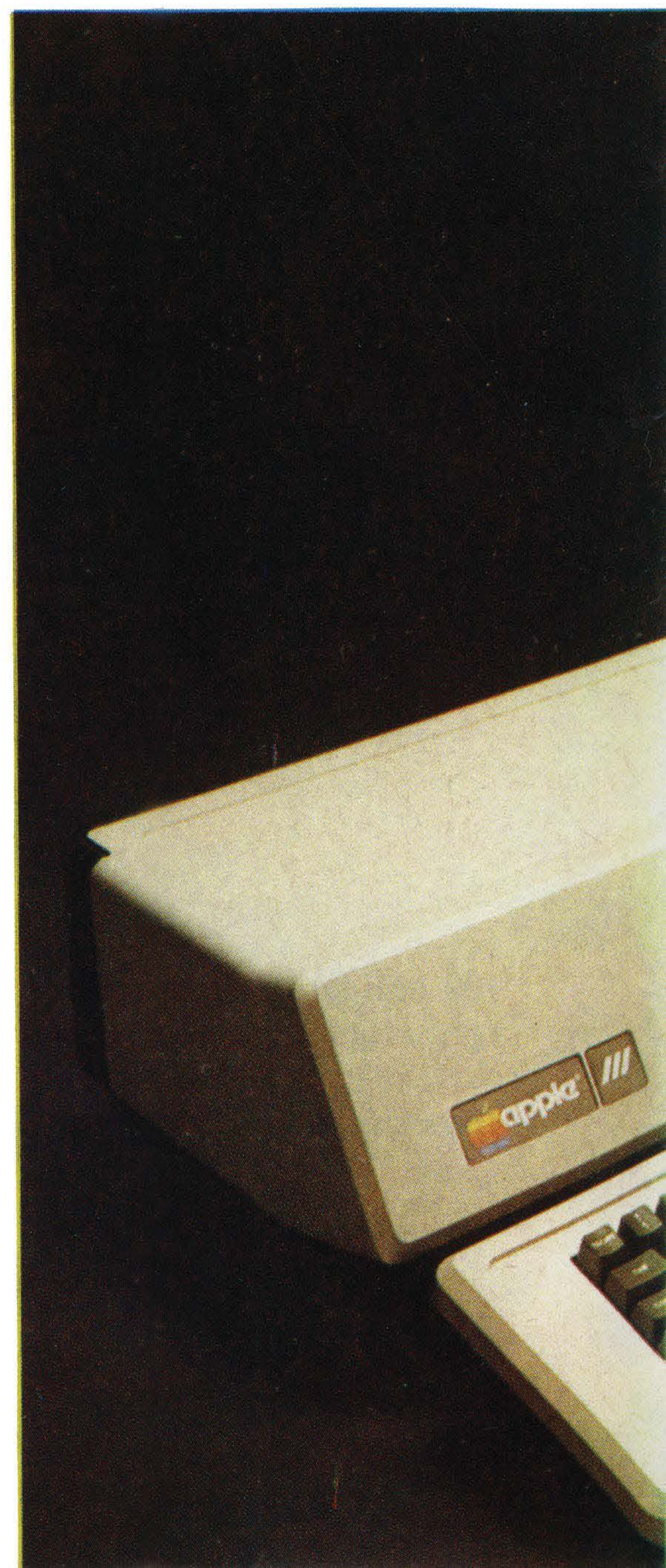
Un'ultima informazione, prima di vedere le caratteristiche principali dell'Apple III: nella presentazione alla stampa specializzata è stata data l'indicazione che il nuovo prodotto sarà disponibile sul mercato europeo entro la fine dell'anno. Per quanto riguarda il mercato italiano, il discorso passa attraverso il distributore ufficiale, che è la IRET di Reggio Emilia.

Vista d'insieme

L'Apple III non si discosta dal suo predecessore: le scelte di fondo realizzate con l'Apple II (tipo di microprocessore, caratteristiche grafiche, portatilità del sistema) sono state mantenute nel nuovo prodotto, che però, ovviamente, presenta alcune caratteristiche peculiari che lo distinguono nettamente dall'altro: tanto per citare le più appariscenti, un minifloppy (capacità 143 kbytes) incorporato nello châssis, e la possibilità di indirizzare fino a 128 kbytes di memoria.
La tastiera, munita di settore numerico separato, e decisamente staccata dallo châssis (a cui comunque è collegata in maniera rigida) tiene conto della comodità dell'operatore, cosa che certo non si può dire del modello precedente. E' rimasta invece invariata la scelta dello schermo video staccato. Ciononostante, l'Apple III pesa circa il doppio dell'Apple II.
Per quanto concerne il sofware di base, l'Apple III è dotato di un sistema operativo chiamato S.O.S. (Sophisticated Operative System), mentre, come linguaggi di programmazione, sono presenti un Basic di 18K, Pascal, FORTRAN IV, ed il PILOT che è un linguaggio indirizzato allo sviluppo di programmi per C.A.I. (Computer Assisted Instruction). Comunque già in questa prima apparizione l'Apple III è accompagnato da: a) un package applicativo denominato Information Analyst Package, comprendente una serie di programmi (VisiCalc III, Mailing List Manager) finalizzati al supporto di attività manageriali - professionali; b) un package di world processing.

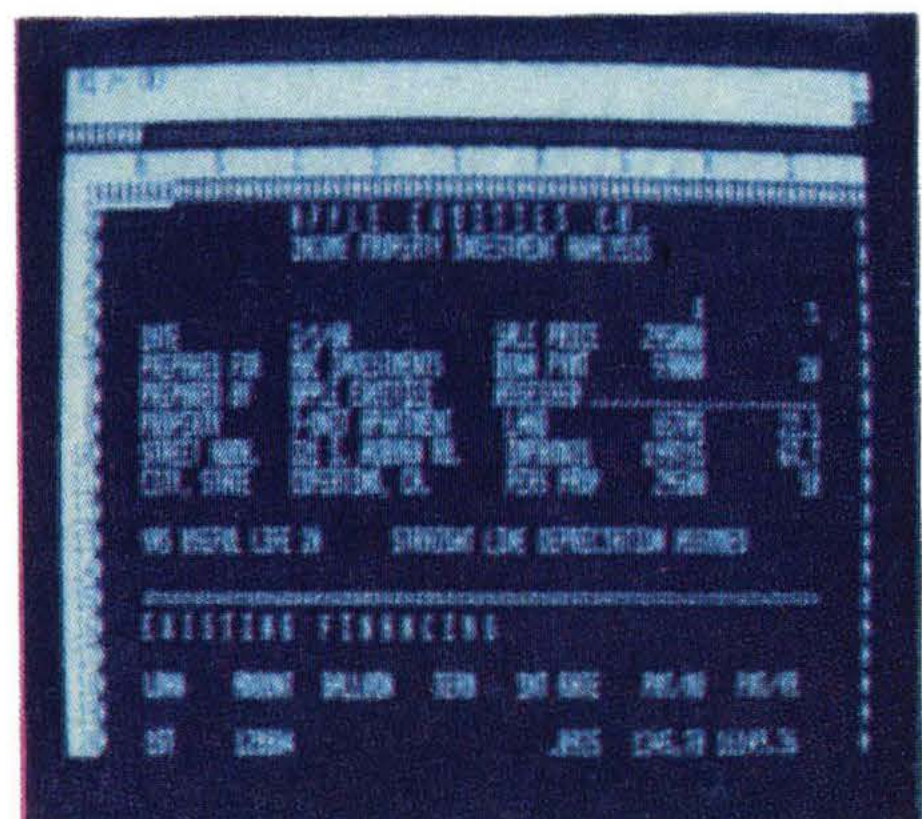
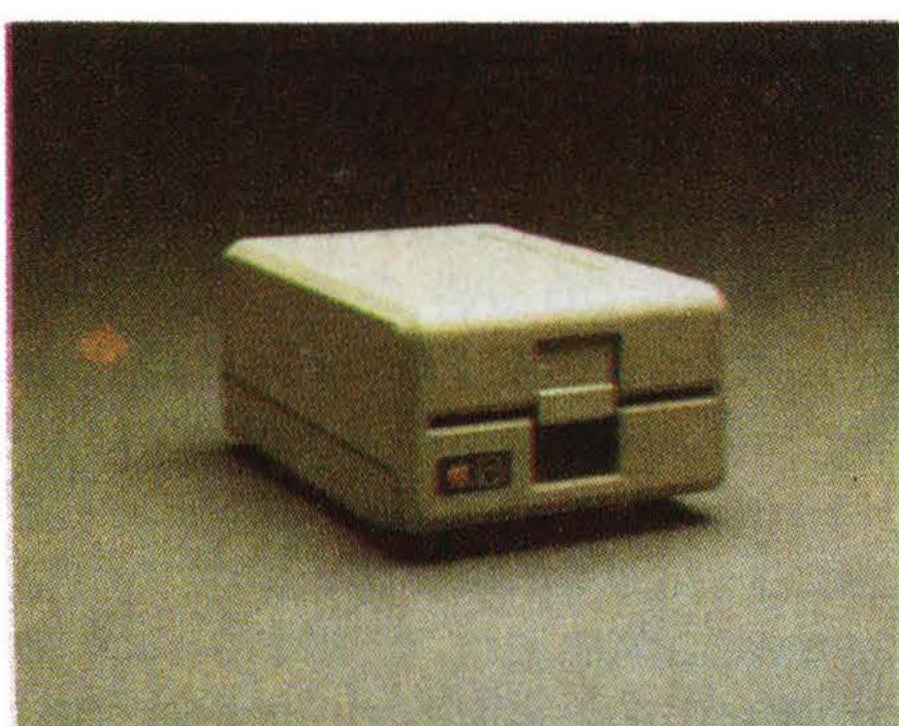
Fermo restando che un discorso di valutazione sulla validità di questi programmi non è lo scopo immediato di questa breve nota, le caratteristiche citate denotano, a mio parere, un'interpretazione ben precisa del fenomeno personal computer da parte del management Apple, l'interpretazione cioé del personal computer come di uno strumento per applicazioni professionali, alcune delle quali stimolate e definite da packages applicativi che fanno del personal computer un prodotto che non necessita di ulteriori interventi. Se a questo elemento si aggiunge il supporto di un linguaggio di programmazione come il PILOT e l'esistenza della Apple Education Fundation, un'organizzazione indipendente a cui partecipano, oltre alla Apple Computer, Inc., anche altre società americane, creata per la promozione e il supporto concreto a nuovi metodi di insegnamento tramite i piccoli elaboratori, possiamo leggere nell'azione dell'Apple, al di là di una semplice politica di autopromozione, un intervento veramente originale nel campo dei personal computer, o comunque, a volere essere prosaici, una ben precisa strategia di marketing in un settore che in un breve lasso di tempo ha visto nascere (anche scomparire) numerose iniziative, e che comunque rimane esposto all'impatto con i grandi dell'EDP quanto più ci si avvicina ai loro settori tradizionali.

Hardware

La soluzione hardware implementata nell'Apple III è originale nella misura in cui si può parlare di originalità in questo settore. Senza dubbio è interessante la conferma del 6502 quale microprocessore ma, qui è la sorpresa, le caratteristiche della scheda di CPU sono state potenziate notevolmente. Tanto per citare i pochi dati comunicati, si ha:
— indirizzabilità di 128 kbytes;
— frequenza di clock uguale a 1,4 MHz (1,8 MHz con video non attivato);
— convertitore digitale-analogico a 6 bit;
— rilocazione dei registri di pagina base;
— rilocazione dello stack.
La prima caratteristica indica la tendenza a portare i dati totalmente in memoria, a vantaggio di una maggiore velocità di esecuzione: in una certa misura la memoria RAM diventa la *copia*

Un'altra mela sull'albero apple: l'APPLE III

quasi integrale della memoria magnetica (l'impostazione classica è quella di vedere la memoria magnetica come un'*estensione* della memoria RAM). Nella versione iniziale comunque l'Apple III è fornito di 96 kbytes di RAM(dinamica), ottenuta con 48 chips di 16 x 1 kbit, per cui, quando diverranno disponibili (commercialmente parlando), chips di 36 x 1 kbit, sarà possibile la piena indirizzabilità di 128 kbytes.

Il convertitore digitale-analogico incorporato è citato giusto per curiosità: comunque può essere utilizzato per sintesi di voce e per l'emissione di suoni.

La rilocazione dello stack e dei registri di pagina base è probabilmente riflessa a livello di S.O.S. (il termine Sophisticated permette di pensare che sia stato fatto qualcosa di più rispetto a quanto solitamente si fa su questo tipo di macchine), e quindi è presumibile che questo abbia un impianto multitask.

Alcuni elementi che erano opzionali nell'Apple II sono stati resi standard nel nuovo modello. Ad esempio la scheda di disk controller è parte integrante dell'Apple III (ed è in grado di pilotare fino ad un massimo di quattro drivers di minifloppy); inoltre è presente l'interfaccia per la stampante termica Apple Silentype (80 caratteri di stampa in modo Text) nonché l'interfaccia seriale RS 232-C. Peraltro sono disponibili quattro posti piastra per schede di I/0. Infine è da citare come parte inte-

grante del sistema un circuito di clock/calendario, utilizzato fra l'altro per datare i files eventualmente aggiornati.

Tastiera e video monitor

La tastiera, praticamente isolata dal corpo del computer per una maggiore comodità di utilizzo, è dotata di 74 tasti, di cui 13 (quelli numerici) duplicati a parte sulla destra per funzioni di entry numerico. Vi sono inoltre quattro tasti di movimento cursore, il tasto di Repeat, il tasto di Alpha-Look, che permette di scegliere fra alfabeto minuscolo e maiuscolo, e infine due tasti funzione ai quali da programma possono essere dati significati particolari. Infine il tasto di Reset è stato posto dietro la zona della tastiera per evitare accidentali operazioni di reset del sistema.

Anche la parte di visualizzazione presenta miglioramenti rispetto alla soluzione adottata con l'Apple II. Ferma restando la scelta di considerare il video un elemento staccato dal resto del sistema (è privilegiata la portatilità della macchina, o meglio la possibilità di inserirla più facilmente in un qualunque ambiente), la funzione di video è affidata ad un monitor da 12 pollici (o maggiore) di tipo RGB (*Red, Green, Bleu*) che consente una visualizzazione di 1920 caratteri distribuiti su 24 linee di 80 caratteri ciascuna. Le lettere dell'alfabeto possono essere minuscole oltre che maiuscole.

Per quanto concerne le caratteristiche grafiche, appare perfezionata quella che è diventata una caratteristica dei sistemi Apple.

I dati comunicati parlano di:

- ultra-alta risoluzione (solo in bianco e nero) e relativa a 560 x 192 punti;
- alta risoluzione (allargata a 16 colori) e relativa a 280 x 192 punti.

Inoltre in bianco e nero è possibile avere 16 toni di grigio.

A parte ulteriori considerazioni su questi dati (ad esempio sull'Apple II in alta risoluzione non è possibile colorare con due colori diversi due punti adiacenti), è da aggiungere solo un fatto di cronaca: il programma dimostrativo che è stato fatto vedere (animazione di figure colorate) è stato di un'efficacia che è raro trovare in macchine della categoria dell'Apple III.

Software

Ancora due parole sul S.O.S (Sophisticated Operative System). Ho già detto che l'impianto sembra essere di tipo multitask. Di fatto è composto di quattro moduli di cui uno, chiamato "Event Management Interface", gestisce le richieste di interruzione scatenate dall'hardware periferico: il sistema evolve in base all'accadimeto di eventi esterni, asincroni rispetto all'attività interna, per cui, fra l'altro, è ottenuta una maggiore velocità di elaborazione.

Gli altri moduli S.O.S. sono:
- il "Module Device", per la gestione di dispositivi di ingresso/uscita byte-oriented (come ad esempio la tastiera) o block-oriented (come il minifloppy);
- il "File System", per l'accesso ai dati da parte dei programmi applicativi dei vari linguaggi;

— il "Memory Managment System", per la gestione delle partizioni di memoria richieste dalle attività installate sul sistema.

Questi brevi cenni giustificano l'enfasi con cui la Apple ha dato comunicazione di questo package; senz'altro il S.O.S. sembra possedere caratteristiche non reperibili su altri personal, per quanto, a livello utente e per applicazioni normali, non è detto che siano evidenti in maniera tangibile dato che, non dimentichiamolo, fine di un sistema operativo è quello di mascherare all'utente i dettagli (più o meno complessi) di operatività di una macchina.

Ultimo elemento da citare è la possibilità dell'Apple III di emulare l'Apple II: in pratica questo vuol dire che i programmi scritti per l'Apple II possono girare sull'Apple III, e quindi che gli investimenti software fatti dagli acquirenti dell'Apple II non sono vanificati passando al nuovo modello. Ovviamente le prestazioni dell'Apple III in questo caso sono identiche a quelle dell'Apple II.

Conclusioni

La comparsa di un nuovo prodotto dell'Apple era data per scontata, quanto meno come prevedibile allineamento con quanto hanno già fatto Radio Shack (con il TRS-80 Modello II) e PET (con il 3032); a questa considerazione va aggiunta l'esigenza per un produttore di sistemi di presentare periodicamente nuovi prodotti. Tutto questo per dire che, se in origine il mercato del personal computer poteva presentarsi per i vari operatori (dal costruttore al venditore) abbastanza oscuro, oggi i prodotti presentati sono frutto di considerazioni meno improvvisate e soprattutto di indicazioni raccolte nel frattempo: l'Apple III è a mio parere una macchina indirizzata ad un mercato professionale.

In quest'ottica non ritengo per nulla riduttiva la scelta di una memoria di massa quale è il minifloppy, in quanto è giusto che nel settore di mercato in cui ci si vuol correttamente porre non vengano fatte pagare prestazioni non necessarie.

Col che si è arrivati al discorso prezzo. I dati comunicati sono puramente indicativi, in quanto fanno riferimento agli U.S.A.

L'Apple III, con 96 kbytes di memoria (versione iniziale), ha un prezzo di 3490 dollari, incluso il driver del minifloppy; il monitor da 12 pollici in bianco e nero 300 dollari; la stampante termica Silentype 595 dollari; il package applicativo "Information Analyst" 430 dollari.

Comunque la Apple anche offre due soluzioni "chiavi in mano": l'"Information Analyst System" (ossia Apple III + monitor + stampante termica + S.O.S. + BASIC + relativo package applicativo), offerto a 4500 dollari, e il "Word Processing System" (come sopra, esclusa la stampante termica, sostituita con una stampante margherita del valore di 3000 dollari, + relativo package di word processing), offerto a 7800 dollari. Questo modo di presentare l'Apple III, non solo computer ma anche macchina specializzata che non necessita di ulteriori interventi, è senz'altro un modo efficace di dare *identità* al personal.

La validità dell'approccio e la consistenza del/dei sistemi non mancheranno di essere valutati più esattamente quando l'Apple III sarà una realtà anche per il mercato italiano. . ■

Stampante termica per l'Apple

L'APPLE II però non si ferma ... È disponibile infatti una stampante termica, la stessa prevista per APPLE III, denominata Silentype che completa di interfaccia per APPLE II, è posta in vendita, dalla IRET di Reggio Emilia, al prezzo di L. 680.000 + IVA.
La Silentype è silenziosa, leggera (Kg 2,7), di minimo ingombro (19,7 x 31,1 x 7) e si presenta come una periferica di grande versatilità.
Questa stampante è in grado di fornire copie di normali testi stampati alla velocità di 40 caratteri al secondo nel formato di 80 caratteri per linea utilizzando una matrice di 5x7 punti.
Rilevante la possibilità di produrre copie di grafici alla velocità di 240 punti al secondo con una lunghezza di linea pari a 480 punti.
La risoluzione orizzontale e verticale di stampa in modalità grafica risulta di 23 punti per centimetro.
La possibilità di stampa in modo grafico rende l'utilizzo, di questa stampante, particolarmente vantaggioso con l'APPLE.
La connessione della Silentype con l'unità APPLE si realizza semplicemente inserendo la scheda di interfaccia in uno degli slot di espansione previsti.
Il set ASCII disponibile è costituito da 96 caratteri e comprende lettere maiuscole, minuscole e codici di controllo per migliorare l'operatività della stampante.
Oltre ai normali comandi di back space, interlinea, ritorno carrello ed avanzamento carta sono stati previsti i comandi per l'attivazione del modo grafico (CTRL/Q), stampa del buffer Silentype (CTRL/F), disattivazione del modo grafico ed interruzione della stampa (CTRL/C).
Esiste anche un comando particolare (CTRL/T) con il quale è possibile *congelare* il contenuto del video, che ha un formato di 40 caratteri per linea, quando la stampante è predisposta per la stampa di 80 caratteri per linea.
Le operazioni di caricamento della carta sono facilitate dall'assenza dei pattini di trascinamento carta e del nastro di stampa.
La Silentype utilizza rotoli di carta termica di circa 24 metri di lunghezza con un formato di centimetri 21,6.
È bene ricordare che tutte le operazioni di stampa possono essere effettuate direttamente oppure tramite software.

di **M. Salvemini**

Premessa

Digitalizzatore, tavoletta digitalizzatrice, tavoletta di input, ed altre, sono tutte espressioni usate nella lingua italiana per tradurre la parola della lingua inglesa *digitizer*. Lunga sarebbe la disquisizione sul termine migliore da utilizzare; per amore di semplicità userò il termine inglese digitizer, ovvero in alternativa di quello italiano *digitalizzatore*.

Nella mia esperienza professionale in Italia non ho notato un notevole sviluppo di questa periferica indispensabile in un sistema grafico automatizzato, sia esso interattivo o non; mi sembra quindi opportuno farne una descrizione dettagliata anche e soprattutto per coloro i quali non l'hanno mai utilizzata.

È ovvio per chi si occupa di computer grafica (ma è facilmente immaginabile anche da chi pur non essendone un utilizzatore, è informato dei suoi sviluppi ed applicazioni) che, mentre è semplice interagire con l'immagine prodotta dal computer, è complicato *immettere* l'immagine (almena la prima) nel computer stesso, per poi processarla. Il problema è quello relativo alla trasformazione di informazioni analogiche (il disegno fatto su di un pezzo di carta) in informazioni digitali (le coordinate dei punti forniti al computer).

Prima che la computer grafica decollasse nel campo dell'informatica, la necessità di introdurre nel computer dati digitali in grande quantità non era stata considerata un grande problema in quanto i dati da immettere erano quasi sempre oggetto di una elaborazione precedente. Quando però (all'inizio degli anni '60) ci si pose il problema di introdurre una notevole massa di dati provenienti non da elaborazioni precedenti ma dalla interpretazione digitale di informazioni analogiche, ad esempio descrizione di una curva di livello in una mappa geografica, si sentì la necessità di automatizzare la procedura di trasformazione dell'informazione analogica in quella digitale e di poter interagire con il sistema grafico mediante una periferica non digitale quale è la tastiera alfanumerica. La nascita del digitizer è recente: la si può fare risalire agli anni '60 (vedi bibliografia allegata).

Alcune nozioni fondamentali

Un digitizer è composto da:
- il piano di lavoro
- lo stilo, puntatore o trasduttore
- la parte di rilevamento della posizione dello stilo
- la parte di hardware interna al digitizer (a monte dell'interfaccia di collegamento)
- l'interfaccia con il computer

— Il piano di lavoro

Un digitizer è costituito da una superficie piana a forma rettangolare, generalmente in materiale plastico, sulla quale è possibile adagiare carte, lucidi etc., di cui a mano libera vengono seguiti linee, contorni, figure. Se il digitizer è collegato ad un video, il disegno viene eseguito con uno stilo o puntatore nella stessa maniera con la quale si

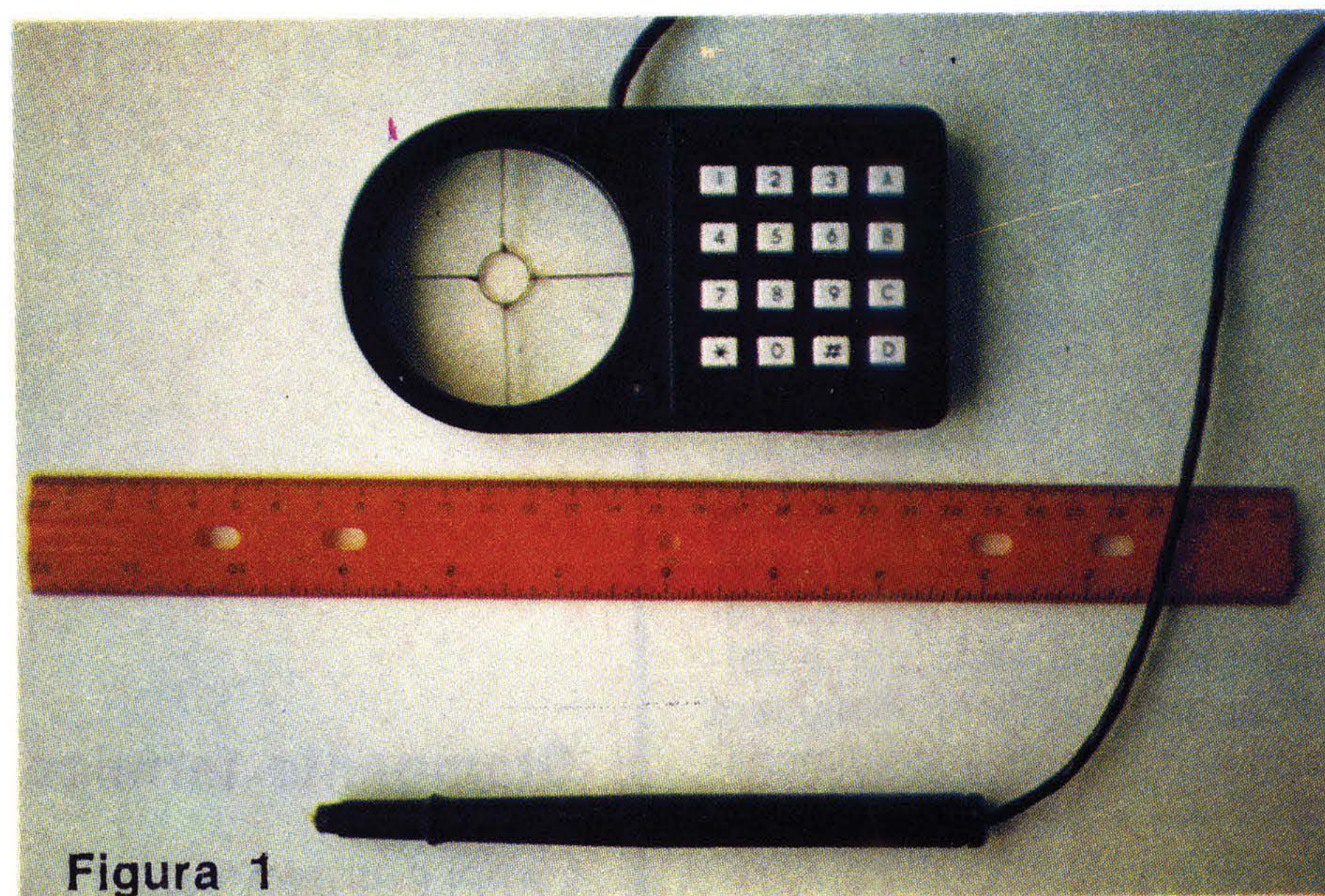



Figura 1

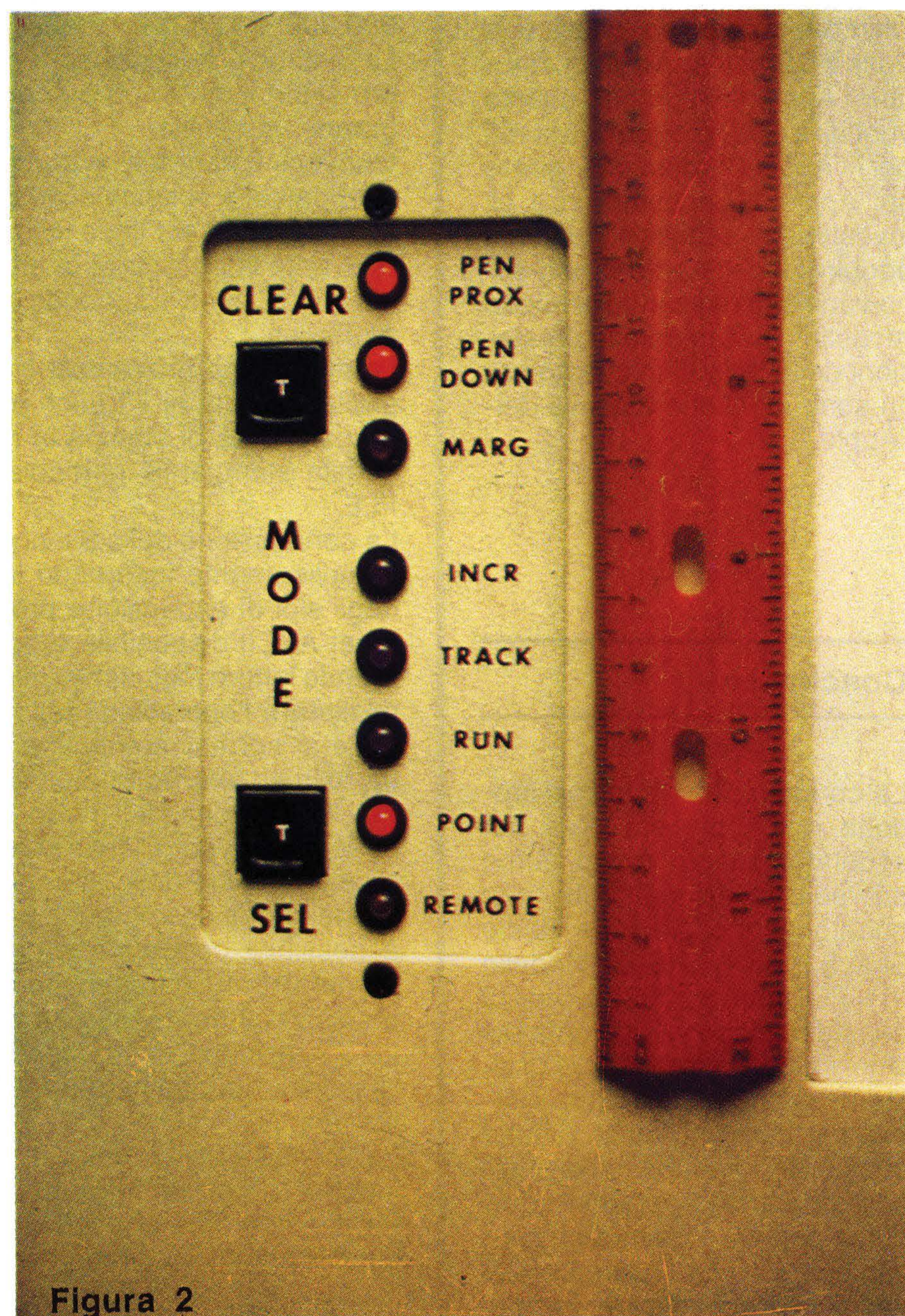



Figura 2

Il digitizer: una periferica per computer grafica

disegna con una penna su un pezzo di carta: il disegno sul video (cioè i dati all'interno del calcolatore) è ottenuto testando periodicamene la posizione dello stilo e riportando le coordinate al sistema che, mediante un software opportuno, unisce poi sul video le coordinate dei punti rilevati.
La consistenza del supporto deve essere funzione della possibilità di interazione tra penna e piano, qualora questa interazione sia parte del sistema di rilevamento della posizione della penna.
Esistono piani in materiali trasluci che permettono di lavorare su trasparenti: sono particolarmente utili per applicazioni specifiche, come lo studio delle radiografie.

— trasduttori di posizione

Lo stilo, puntatore o trasduttore è parte integrante del digitizer ed è particolarmente importante poichè è il mezzo con il quale avviene l'interazione tra operatore e sistema.
Si chiama generalmente *stilo* una penna fornita di un interruttore tale che al premere della penna sul piano un circuito venga chiuso. La penna o stilo permette di avere per ogni punto tre tipi di inormazioni: due coordinate del punto e l'indicazione dell'apertura o chiusura del circuito di invio delle coordinate al computer.
Il *puntatore* invece è un componente in grado di inviare più di tre informazioni alla volta, in quanto è fornito di tastiera: permette quindi di interagire con il sistema in modo vario. Risulta peraltro molto più preciso dello stilo sia per facilità di puntamento sia per migliore definizione del riferimento di puntamento, il quale a volte a volte è fornito di lente di ingrandimento per facilitare l'operatore.
L'uso della penna e/o del puntatore è dettato dalle necessità particolari dell'utente. È ovvio che il puntatore è più versatile del semplice stilo e in particolare risulta utile qualora l'operatore, oltre a normali operazioni di digitalizzazione, debba fare operazioni "intelligenti" quale quella di immettere le coordinate di inizio di un'area, di fine, o voglia interagire con il sistema: premendo un tasto si introducono le coordinate del punto rilevato fornite dal proprio indirizzo.
Inoltre è opportuno ricordare che tramite il puntatore, se si ha un digitizer a disposizione, si possono eseguire direttamente operazioni quali il calcolo di un'area, di una lunghezza di un volume, etc.

— rilevamento della posizione

Sono stati trovati vari metodi per eseguire le operazioni di rilievo delle coordinate sul piano di lavoro. I più comuni sono:

- un campo elettrico è generato sulla superficie di un mezzo conduttore cosicchè viene rilevato il potenziale dallo stilo stesso;
- un campo magnetico è generato a quindi rilevato da una induttanza nello stilo cosicchè l'intensità del voltaggio indotto è funzione unica della posizione della penna;
- una scintilla è periodicamente generata dalla punta della penna:acusticamente la scintilla produce un suono acuto che si disperde in tutte le direzioni e che è ricevuto da alcuni microfoni lineari posizionati su due lati adiacenti del piano di lavoro. Misurando il tempo di propagazione dell'impulso di suono, si ottiene la misurazione della distanza dello stilo dai due lati che rilevano il suono e quindi la posizione dello stilo stesso;
- un campo emesso dallo stilo è ricevuto da una griglia posizionata sotto la superficie della tavoletta. La posizione dello stilo è determinata dal conto delle meglie della griglia combinata con la fase: questa comparazio-

Figura 1 - La foto mostra due tipi di puntatori o trasduttori: a stilo ed a cursore. In quello a stilo si può vedere l'interruttore alla punta della penna stessa; per quello a puntatore invece il circuito viene chiuso e l'informazione inviata alla pressione di uno qualunque dei tasti presenti su di esso. Esistono generalmente alcuni problemi nell'usare lo stilo invece del puntatore: l'handicap più grande lo si trova nella difficoltà di rilevare un punto, poichè è complicato vedere il punto una volta posizionata la penna nel suo intorno.
Mediante il puntatore il posizionamento sul punto da rilevare è più agevole e preciso. Esiste poi la comodità di poter inviare al computer più informazioni, oltre alla posizione della penna, mediante la piccola tastiera sul cursore. È possibile operare sia con la penna che con il cursore mediante menu, cioè aree del tavolo digitalizzatore riservate alla possibilità di far eseguire operazioni al computer al solo posizionare la penna o il puntatore nell'area prescelta. (v. Figura 3).

Figura 2 - In alcuni digitizers si può selezionare il modo operativo mediante una ulteriore tastiera posta sul piano di lavoro vicino all'operatore. Su di essa possono anche apparire avvertimenti per l'operatore per quanto concerne lo stato del puntatore. Nella figura la spia MARG si accende se il puntatore è al di fuori dell'area di lavoro. Come è stato già detto, esistono vari modi di lavorare del digitizer. In questa immagine essi sono rappresentati nella stessa, tastierina che fornisce informazioni all'operatore sullo stato della penna. Sui digitizer evoluti sono generalmente disponibili più "modes" operativi. Cerchiamo di esaminare quelli mostrati in figura. Il più elementare è il "Point Mode": ogni volta che la penna viene premuta sul tavolo (o si preme un pulsante del cursore) si effettua la digitalizzazione del punto centrato. Questa è la tipica rilevazione per punti. Il secondo è il "Run Mode", in cui la digitalizzazione è continua ed inizia non appena il trasduttore raggiunge la zona di prossimità del tavolo. La cadenza di rilevazione è fornita automaticamente da un apposito circuito e può essere variata a piacere. Abbiamo poi il "Track Mode", assai simile al precedente. Anche qui si ottiene una digitalizzazione continua che si effettua, però, solo quando il microinterruttore del trasdutore è attivato. Anche qui la cadenza è fornita elettronicamente. L'"Increment Mode" invece effettua la digitalizzazione continua solo quando il trasduttore è in movimento. Infine il "Remote Mode": la rivelazione è continua ma viene attivata non più dal microswitch ma da una sorgente esterna (generalmente un impulso di richiesta dati proveniente dall'elaboratore).

NANOCOMPUTER®

UN COMPUTER PER IMPARARE TUTTO SUI COMPUTER.

In questi ultimi anni, l'eccezionale diffusione dei microprocessori nell'industria e nella vita di tutti i giorni ha aumentato fortemente la richiesta di persone in grado di operare professionalmente nel settore.

La SGS-ATES, uno dei maggiori produttori di microprocessori da sempre in primo piano nel loro supporto in Europa, ha fatto fronte a questa esigenza realizzando il NANOCOMPUTER, un sistema didattico professionale e completo.

<u>Insegnamento e apprendimento: due facce dello stesso problema.</u>

Su questo concetto è basato il sistema didattico NANOCOMPUTER in cui la SGS-ATES ha riversato una lunga esperienza sistemistica e produttiva, realizzata preparando i suoi tecnici e ricercatori ad altissimo livello.

Il NANOCOMPUTER è un sistema didattico integrato e modulare. È formato da un potente microcalcolatore con il microprocessore Z80 prodotto in Italia dalla SGS-ATES, e da un insieme completo di sussidi educativi: libri di testo Nanobook® in italiano e nelle principali lingue europee, manuali tecnici, kit per esperimenti.

La concezione modulare permette al NANOCOMPUTER di crescere insieme allo studente, in un processo di apprendimento attivo fondato sul continuo dialogo tra la macchina e lo studente.

NBZ80-S. Scheda base, scheda per esperimenti, miniterminale, contenitore-alimentatore, kit di fili, Nanobook 1 e 3, manuale tecnico.

SGS-ATES Componenti Elettronici S p A , Via C. Olivetti 2, 20041 Agrate Brianza, Tel. (039) 65551 • DIREZIONE COMMERCIALE ITALIA, Via Correggio 1/3, 20149 Milano, Tel. (02) 4695651 • UFFICI VENDITA: Firenze, Tel. (055) 4377763 • Milano, Tel. (02) 4695651 • Roma, Tel. (06) 8392848 • Torino, Tel. (011) 531167 • PUNTI DI VENDITA: Via Larga, ang. Via Brolo/Via Verziere, 20122 Milano, Tel. (02) 8690047 • Via S.

Per queste caratteristiche, il sistema NANOCOMPUTER è particolarmente adatto non solo all'apprendimento a scuola, sotto la guida di un insegnante, ma anche per chi voglia individualmente prepararsi a questa nuova professione.

Il sistema NANOCOMPUTER: un sistema modulare. Il NANOCOMPUTER, studiato espressamente per impieghi didattici, riunisce in sé un'elevata rigorosità di concezione e un'estrema flessibilità, essenziali in un processo di apprendimento teorico e sperimentale al contempo. Nella sua versione più semplice, NBZ80-B, il NANOCOMPUTER permette anche allo studente senza conoscenze specifiche di impadronirsi delle tecniche di programmazione dei microprocessori.
Con la versione NBZ80-S lo studente viene introdotto anche nelle tecniche di interfacciamento di un microprocessore con il mondo esterno e nei problemi di interazione tra hardware e software.

NBZ80-B. Scheda base, miniterminale, contenitore-alimentatore, Nanobook 1, manuale tecnico.

È possibile, attraverso un kit di espansione, passare dalla versione NBZ80-B alla NBZ80-S. In tal modo ogni studente può scegliere, graduandolo nel tempo, il livello di apprendimento più consono alle proprie esigenze.
L'NBZ80-S è a sua volta ulteriormente espandibile per consentire l'approfondimento di un linguaggio ad alto livello, il Basic, soprattutto nelle sue interazioni con l'hardware.

NBZ80-HL. Come NBZ80-S con 16k byte di RAM, tastiera alfanumerica con interfaccia video, 8k ROM di Basic su scheda addizionale, libro Basic Programming Primer, monitor TV (opzionale).

Desidero ricevere gratuitamente maggiori informazioni su:
☐ sistema NANOCOMPUTER®
☐ corsi sullo Z80 con l'utilizzo del NANOCOMPUTER®

NOME ____________ COGNOME ____________

INDIRIZZO ____________

PROFESSIONE ____________

Inviare a: SGS-ATES
Componenti Elettronici S p A
Via C. Olivetti 2-20041
Agrate Brianza, tel. (039) 65551

SGS ATES

BIT

Quintino 29/C, 10121 Torino, Tel. (011) 531267 • DISTRIBUTORI: CID, Roma, Tel. (06) 6383979 • DE DO Electronic Fitting & C. S.a.s., Tortoreto Lido (Te), Tel. (0861) 78134 • DISELCO S.p.A., Milano, Tel. (02) 3086141 • FANTON ELECTRONIC SYSTEM S.r.l., Padova, Tel. (049) 654487 • FANTON BOLOGNA S.r.l., Bologna, Tel. (051) 357300 • AGENTE ESCLUSIVO PER LA SCUOLA: GB PARAVIA & C. EDITORI, Torino, Tel. (011) 779166.

Il digitizer

Figura 3 - La figura rappresenta un tipo di menu, abbastanza generale, che può essere creato su di un digitizer. La comodità di disporre di un menu di tale genere è che non occorre di volta in volta approntare un software che esegua le operazioni che nel menu stesso vengono contemplate. È necessario invece di volta in volta posare il puntatore su una della caselle, e il digitizer si configura per accettare i comandi o fare eseguire le operazioni selezionate. Da qui la parola menu, quasi che noi puntando il dito su una lista di un ristorante sulla quale appaiono le foto dei piatti dicessimo: "Voglio questo o quello". Per introdurre un cerchio e quindi farlo disegnare sullo schermo basterà allora immettere le coordinate dal centro ed il raggio dello stesso, e così via per altre figure o, come si dice in gergo, per altre facilities. Non esiste ovviamente un menu che vada bene per tutte le applicazioni di computer grafica, anzi il fatto di disporre di un menu personalizzato è una caratteristica propria di un sistema preparato ad hoc. Esistono menu preparati per il disegno meccanico, per la progettazione civile etc.

Figura 4 - Il digitizer con il quale sono stati eseguiti gli esempi riportati di seguito è completamente aperto. È interessante notare che i 64 conduttori che giacciono sulla parte interna del piano di digitalizzazione e che l'operatore generalmente non vede. Il board aperto a libro è quello sul quale si innesta il collegamento dell'interfaccia che trova posto in uno degli slots dell'Apple. Altri digitizer sono più complessi, non tanto come funzionamento ma come costruzione. Particolare attenzione infatti viene riposta nella costruzione del piano di lavoro che deve avere caratteristiche di resistenza, inerzia ed affidabilità, specie se nel suo interno, come spesso avviene, trova posto una griglia di conduttori per il rilievo delle coordinate. Il piano di lavoro poi, nel caso che si voglia lavorare con supporti traslucidi cioè con fonte di luce diretta posta dietro il tavolo, devono avere anche caratteristiche tali da non far vedere all'operatore la griglia o in genere i conduttori affogati nel piano di lavoro e permettere invece una buona luminosità del piano stesso. A detta di alcuni produttori di digitizer, il cui prezzo generalmente cresce con il quadrato del lato, il costo del digitizer (specie di quelli professionali) è in gran parte imputabile al costo del materiale ed alla tecnologia impiegata nella costruzione del piano di lavoro, oltre ovviamente ai sistemi di controllo dei quali deve essere fornito per la taratura e l'allineamento.

"	!	'	\	$	^	[	]
?	_	&	%	>	<	@	#
:	=	*	;	/	,	(	)
W	X	Y	Z	SPACE	+	-	.
O	P	Q	R	S	T	U	V
G	H	I	J	K	L	M	N
8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7

BEGIN SYMBOL STRING	REPEAT THE CURRENT SYMBOL STRING		
ENTER THE SYMBOL HEIGHT MEASUREMENT	WRITE NEXT SEQUENTIAL BLOCK ADDRESS		
USE THE CURRENT X,Y POINT	MEASURE THE SHADING DISTANCE		
SELECT PEN NUMBER	BEGIN/END SHADED FIGURE DEFINITION	ENTER THE GRID SIZE MEASUREMENT	
ENTER THE RADIUS MEASUREMENT	BEGIN/END FIGURE DEFINITION	REPEAT THE CURRENT FIGURE	
PLOT A CIRCULAR ARC L.T. 180 DEG.	DASHED LINE MODE	ENTER THE DASH LINE LENGTH AND SPACE	
PLOT A FULL 360 CIRCLE	ARROWHEAD MODE	SMOOTH CURVE MODE	
BEGIN COORDINATE STRING	CLOSE COORDINATE STRING	STRAIGHT LINE MODE	

Figura 3

Figura 4

ne indica la posizione dello stilo.

Esistono comunque altri tipi di digitizer. Essi sono generalmente costruiti per scopi particolari: a titolo informativo si possono ricordare i digitizer che al posto dello stilo o puntatore recepiscono il tocco della mano umana, sistema questo particolarmente interessante per utilizzazioni che prevedono utenti non specializzati, o in situazioni particolari. Altri affascinanti digitizer che sono quelli tridimensionali, che riescono a descrivere interamente un oggetto tridimensionale.
Esistono poi periferiche che, pur essendo plotter, hanno la possibilità di trasmettere i dati in input funzionando come digitizer.

Prima di entrare nello specifico della periferica è necessario distinguere tra digitizer intelligenti e non. L'uso di microprocessori in queste periferiche e la costruzione di sistemi completi ha fatto si che alcuni digitizer siano in grado non solo di trasformare informazioni analogiche in digitali, ma di compiere operazioni matematiche e logiche a monte dell'immissione del dato digitale nel computer ospitante: questi sono detti i digitizers intelligenti. Ovviamente sono i più

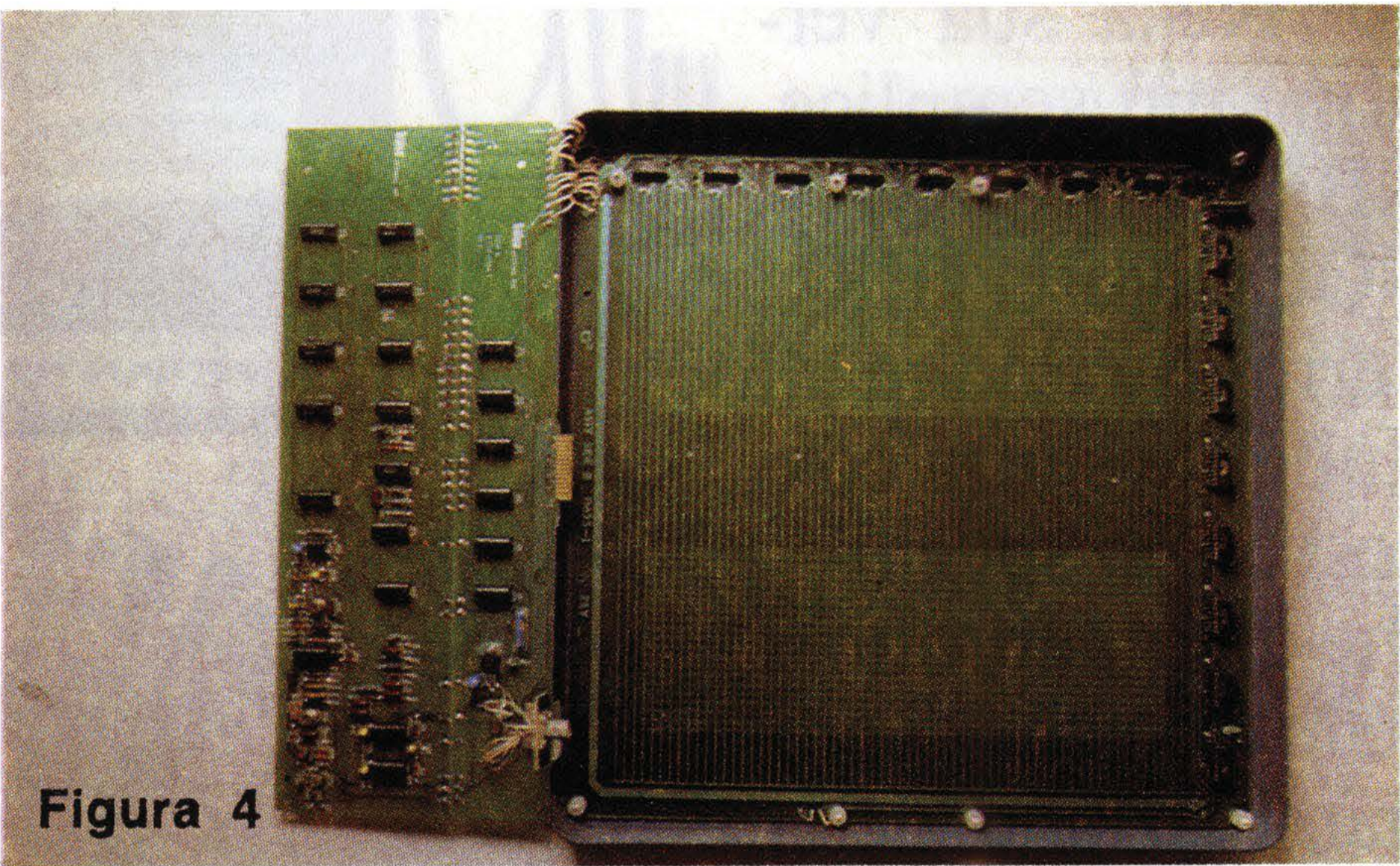
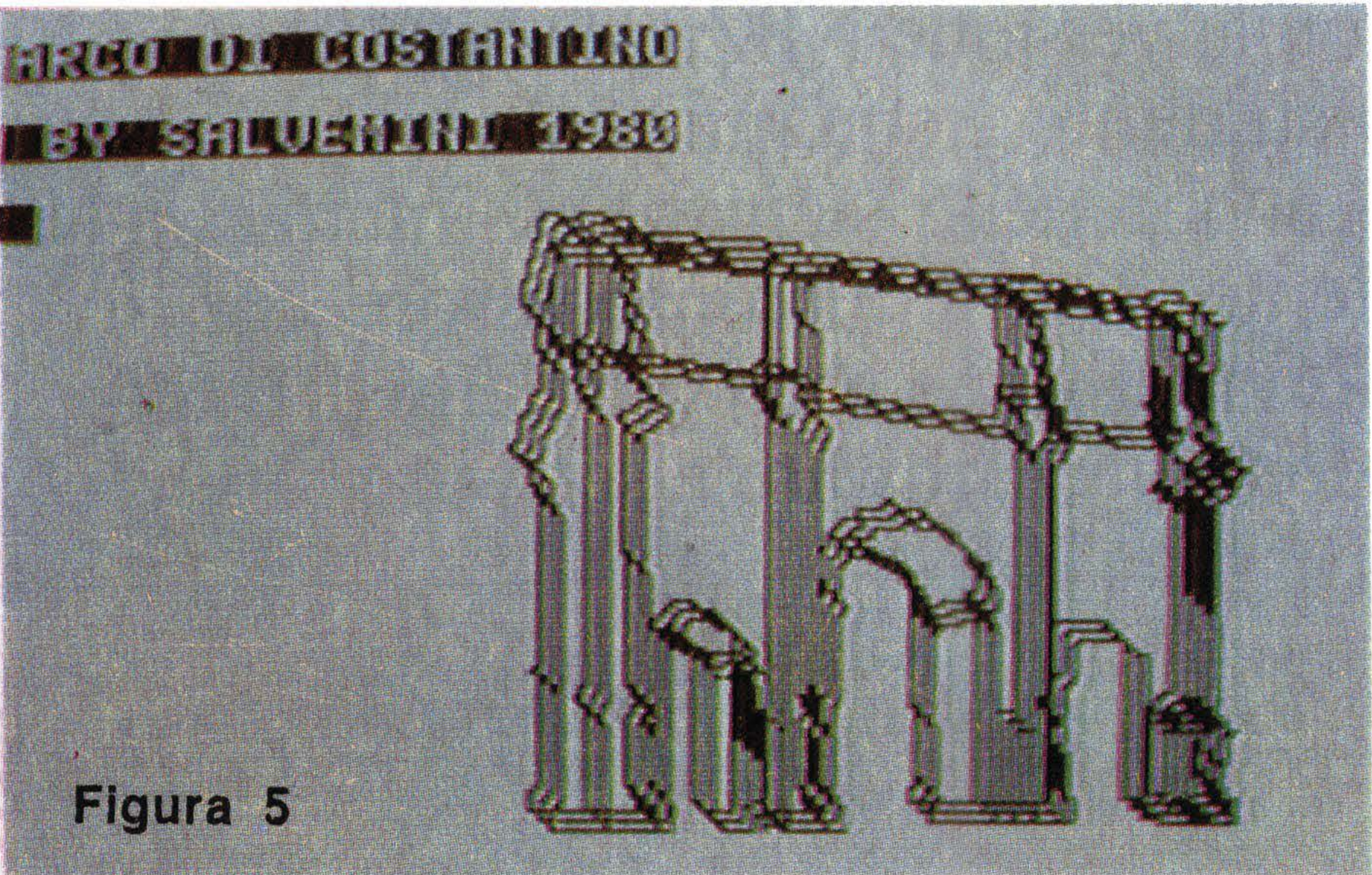



Figura 5

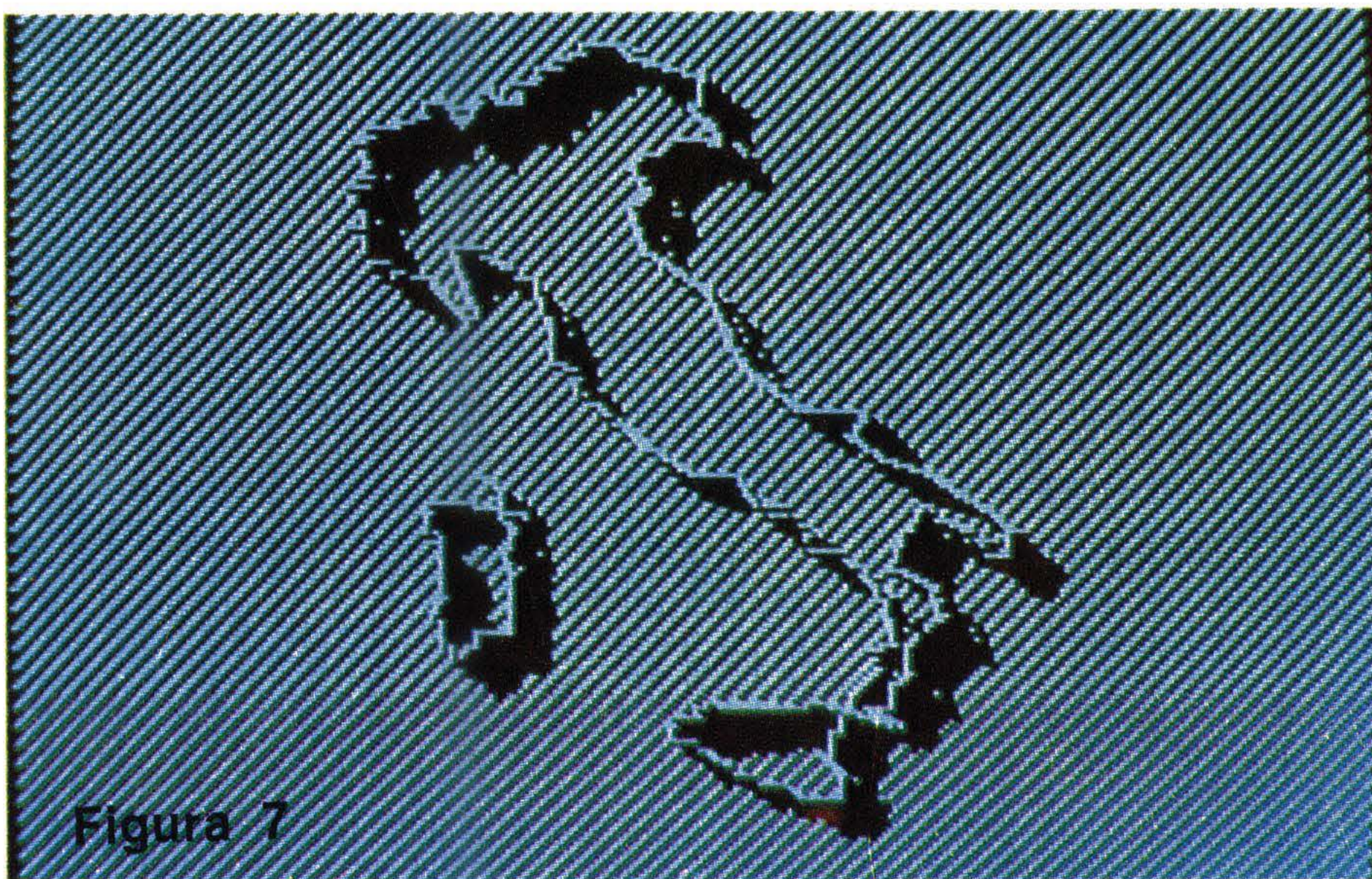
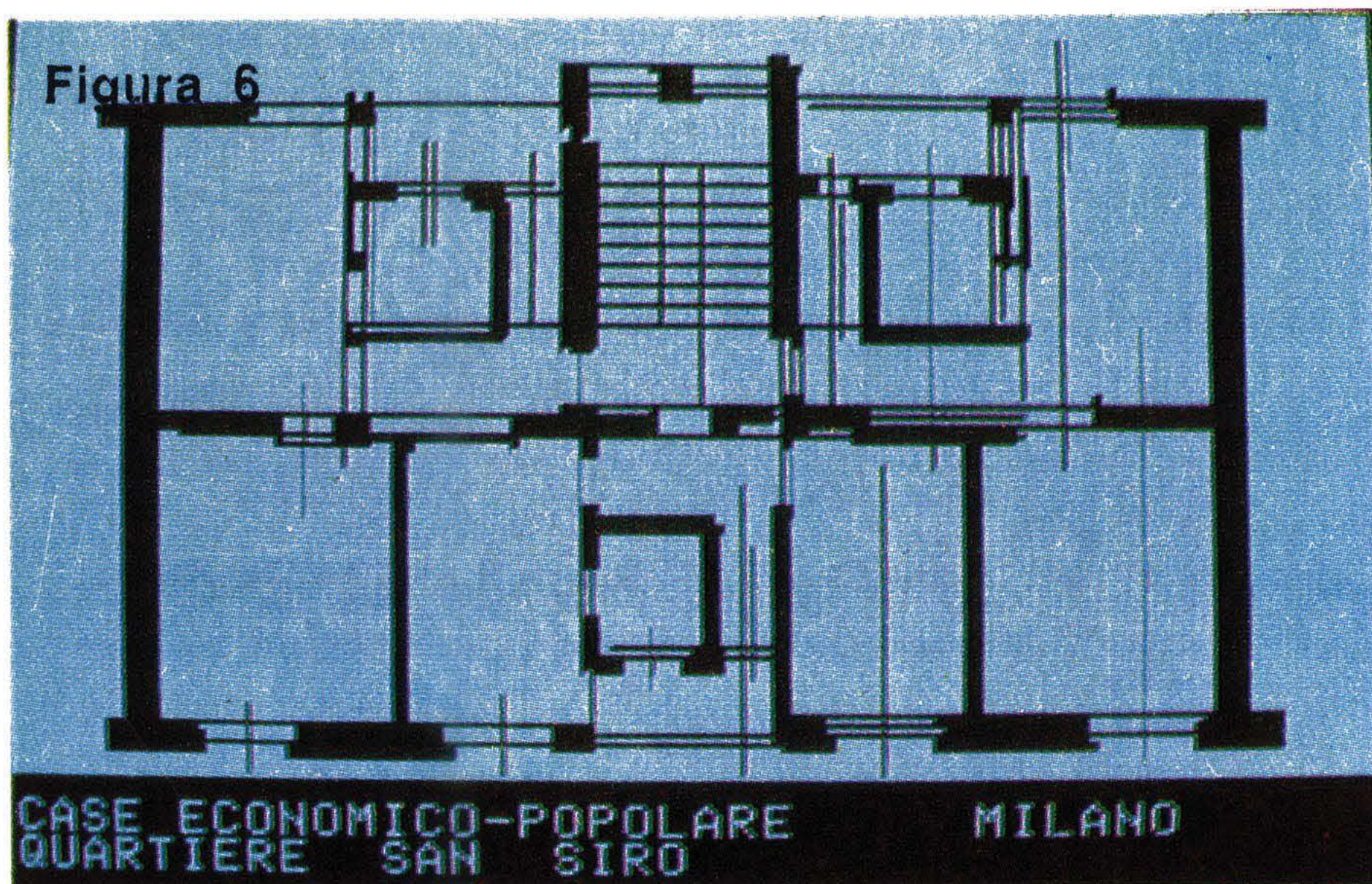



Figura 5 - Riproduzione eseguita mediante digitizer di un monumento, digitalizzando una cartolina postale. Il software per la restituzione del disegno è stato poi opportunamente elaborato in modo tale da ottimizzare la rappresentazione in termini di visibilità del disegno e di gradevolezza dell'immagine.

Figura 6 - La figura rappresenta una applicazione un pò più interessante ed utile della precedente. Da una mappa di una casa di abitazione dell'inizio del secolo sono state tratte informazioni digitali che, opportunamente assemblate, hanno fornito l'immagine che si vede. È opportuno che, una volta rappresentata ed immessa nel calcolatore sotto forma digitale, l'immagine possa essere manipolata come meglio si crede cambiando scala, ruotandola, facendo calcolare le quadrature dei vari vani, etc. Nella progettazione dell'edilizia ed in generale nelle discipline dell'ingegneria e dell'architettura, il digitizer si è rilevato come una periferica molto utile. Il suo sviluppo in questo ambito è destinato a crescere.

Figura 7 - Output grafico di un programma per la mappazione di mappe geografiche: in questo caso viene rappresentata l'Italia. Con qualche accorgimento grafico, già illustrato su BIT è possibile fornire l'immagine della terza dimensione, ma, ovviamente, solo a livello di trick. I punti che formano la figura sono stati rilevati mediante il digitizer, ma con un pò di pazienza possono essere anche immessi a mano.

costosi di quelli normali, ma le funzioni che svolgono si possono rivelare di particolare utilità.

Un esempio pratico

Uno dei digitizer da me usati negli ultimi tempi ben si presta a costituire un esempio applicativo efficace. Si tratta di un digitizer da 11 pollici di piano di lavoro, fornito di propria interfaccia con Apple II: questa interfaccia ha un connettore da 50 pin che può essere allocato in uno qualunque degli slots consentiti dal computer. Dall'interfaccia esce il cavo che si collega direttamente con il digitilizzatore: la tensione al digitizer è fornita tramite l'interfaccia stessa (+ 12 V a 100 mA). Il digitizer invia all'interfaccia 12 bits di dati indicanti la posizione della X ed altrettanti per la Y, un bit che indica se la penna è Up o Down ed un bit che indica se la coordinata inviata è la X o la Y: in conclusione dal digitizer vengono inviati in totale 14 bits per indicare la coordinata X, ed altrettanti per Y.

L'interfaccia trasforma quanto sopra in quattro bytes (1 byte = otto bits) che, una volta allocati nella memoria del computer, sono elaborati dal programma per eseguire le operazioni desiderate.

Per effettuare l'input dei dati dal digitizer nel computer è necessario scrivere in linguaggio macchina il driver d'input: in pratica si tratta di operare una procedura di inizializzazione che in questo caso è stata realizzata mediante il linguaggio Basic con comandi di POKE.

Mi sembra inutile riportare qui le pochissime linee di programma necessarie all'inizializzazione perchè esse sono funzione della macchina presa in considerazione e sono riportate nel manuale della stessa. Basti dire che a seconda della posizione dell'interfaccia negli slots, variano le allocazioni di memoria da andare a leggere.

Le informazioni trasmesse dal digitizer al computer sono tre, il valore della coordinata X, quello della coordinata Y e lo stato della penna (se è Up o Down). In questa apparecchiatura particolare la penna contiene una piccola bobina, posizionata nella punta, che è guidata da un amplificatore AC.

La bobina può essere considerata come il primario di un trasformatore che ha il secondario all'interno dell'area della superficie di lavoro.

La superficie di lavoro possiede 64 conduttori disposti parallelamente l'uno all'altro; la lunghezza di ognuno è di 12,8 pollici, ma poichè questa lunghezza contiene 0,9 pollici di margine su ogni lato, di fatto si ha una superficie di lavoro di 11 pollici quadrati. I conduttori, su di un lato, sono connessi elettricamente tra loro così da assicurare un ritorno comune; sull'altro lato sono connessi in modo tale che ogni conduttore è selezionato elettronicamente e quindi connesso ad un secondo bus.

Quando la penna è posta in prossimità della superficie di lavoro la corrente indotta produce un segnale nelle linee conduttrici che percorrono la superficie stessa, in maniera inversamente proporzionale alla distanza della penna: la posizione relativa della penna rispetto ai conduttori e la direzione dell'asse di scansione sono determinate mediante l'angolo di fase del segnale indotto rispetto

Il digitizer

Figura 8 - Listing di un programma che permette di calcolare l'area di una qualunque figura una volta immessi i vertici di essa. Occorre avere l'accortezza, qualora si usi il programma di immettere come ultimo punto, della figura lo stesso che è stato immesso per primo, in tal modo il programma capisce che la figura è chiusa e che è terminato l'input dei dati. Il programma è stato scritto su di un particolare personal, l'Apple II, che ne permette la visualizzazione su schermo, ma eliminando la subroutine grafica può funzionare su qualunque personal fornito di BASIC. Le limitazioni sono state imposte ai valori di input delle coordinate sono dovute alla possibilità di disegno sull'Apple senza ulteriori algoritmi di trasformazione delle coordinate; tale limitazione si può togliere usando altri computers ovvero introducendo fattori di scala.

al segnale originale. Il voltaggio di questo segnale, inviato ai circuiti analogici, permette di individuare la posizione della penna: la risoluzione realizzata è di 200 punti per pollice (e quindi di 2200 per asse), cioè in pratica di 0,1 mm. L'acquisizione delle coordinate è fatta alternativamente per l'asse X ed Y; dopo che le coordinate sono state trasferite ai circuiti di output, è usata la logica della penna (nei termini di penna Up-e/o penna Down) a seconda che non si voglia o si voglia digitalizzarle ed inviarle come dati al computer ospitante.

Calcolo di un'area

Un programma che può anche essere usato senza digitizer, ma che ben rappresenta l'opportunità di avere uno strumento di questo tipo quando si lavora ad esempio per il calcolo di aree, è quello del calcolo di un'area qualsiasi (v. figura 8). Il calcolo di un'area qualsiasi può essere fatto usando la formula:

$$area = 1/2\,(X_1Y_2 + X_2Y_3 \ldots + X_{n-}Y_n + -Y_1X_2 - Y_2X_3 \ldots - Y_{n-1}X_n - Y_nX_1)$$

Il programma permette di ottenere insieme al disegno, sul video, della figura digitalizzata l'area della stessa.
Esso può essere usato anche senza digitizer, basta usarlo come si è mostrato ed introdurne da tastiera i valori X e Y di ciascuno dei punti precedentemente rilevati: se dovete fare spesso questo lavoro allora vi è chiaro perchè è comodo usare un digitalizzatore.

Conclusione

Da quando il digitizer è entrato come periferica fondamentale nella computer grafica, molte sono state le sue applicazioni; non tutte molto note ai non addetti ai lavori ma tutte egualmente affascinanti. Citiamo la metereologia, la lettura di foto aeree e di dati rilevati da satelliti, l'analisi ortodontica, le applicazioni biomedicali, la topografia, la pianificazione urbanistica, la mappazione e l'interazione per il pronto intervento, il C.A.D. (Computer Aided Design).

È sembrato opportuno allegare

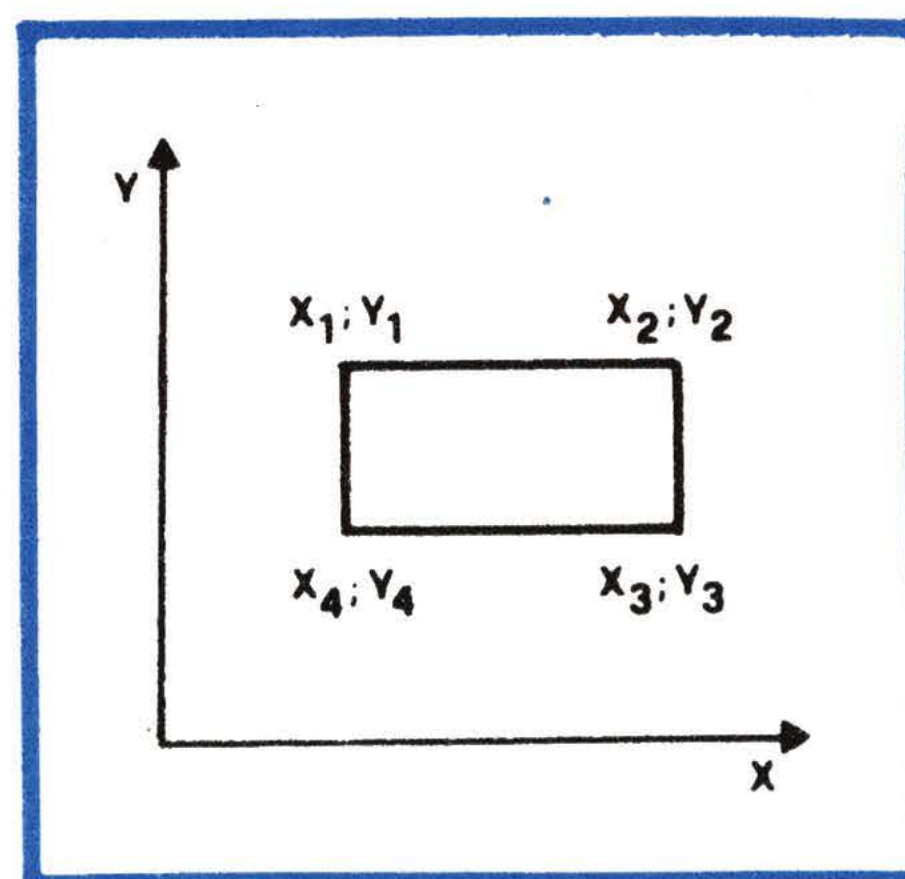
```
        ************************************************
        *                                              *
        *  PROGRAMMA PER IL CALCOLO DI UNA FIGURA      *
        *                                              *
        *              MAGGIO 1980                     *
        ************************************************


1  REM  ***** PROGRAMMA SCRITTO DA MAURO SALVEMINI ***
      *****
2  REM  **'** ROMA MAGGIO 1980 ********
3  REM  ** ** PER LA RIVISTA BIT ********
10  REM  ***** QUESTO PROGRAMMA PERMETTE DI CALCOLARE
      L'ARE. DI QUALUNQUE POLIGONO ***********
20  REM  ***** OCCORRE IMMETTERE I DATI DA TASTIERA***
      *******
30  REM  ***** IL PROGRAMMA E' STATO SCRITTO PER UN AP
      PLE 2 MA TOGLIENDO LA SUBROUTINE GRAFICA SI PUO'S
      CRIVERE PER QUALUNQUE COMPUTER FORNITO DI LINGUAG
      GIO BASIC"
35  CLEAR
40  DIM XCO(300),YCO(300):COUNT = 1
50  REM  ******ISTRUZIONI DI INPUT DELLE COORDINATE X
      ED Y******
59  INVERSE
60  HOME : VTAB 3: HTAB 5: PRINT "CALCOLO DELL'AREA DI
       UN POLIGONO": NORMAL
61  VTAB 5: HTAB 3: PRINT "L'ORIGINE DEL RIFERIMENTO E
      ' IN ALTO": VTAB 6: HTAB 3: PRINT "A SINISTRA NEL
      LO SCHERMO"
65  LET FLAG = 0
70  VTAB 10: HTAB 10: PRINT "PUNTO #";COUNT: VTAB 13: HTAB
      10: INPUT "COORDINATA X =";XCO(COUNT)
80  VTAB 16: HTAB 10: INPUT "COORDINATA Y =";YCO(COUNT
      )
81  REM  ***** ATTENZIONE PER CHIUDERE IL POLIGONO IMM
      ETTERE LE COORDINATE DEL PRIMO PUNTO******
85  GOSUB 1000
86  REM  ***** QUALORA NON SI DESIDERA IL DISEGNO DELL
      A FIGURA ,NON ESSENDOCI LIMITE PER I VALORI DELLE
       COORDINATE,RIMUOVERE LA SUBROUTINE A LINEA 1000*
      *****
87  IF FLAG = 1 THEN  GOTO 65
90  IF COUNT = 1 THEN  GOTO 100
95  GOTO 110
100  LET XFIRST = XCO(COUNT): LET YFIRST = YCO(COUNT)
110  REM  *****CONTROLLO DELLA CHIUSURA DEL POLIGONO**
      ****
115  IF COUNT = 1 THEN  GOTO 150
120  IF XCO(COUNT) < XFIRST + 1 AND XCO(COUNT) > XFIRS
      T - 1 THEN  GOTO 140
130  GOTO 150
140  IF YCO(COUNT) < YFIRST + 1 AND YCO(COUNT) > YFIRS
      T - 1 THEN  GOTO 200
150  LET COUNT = COUNT + 1
155  GOTO 70
200  HOME : VTAB 15: HTAB 10: PRINT "IL POLIGONO E' ST
      ATO CHIUSO"
210  VTAB 18: HTAB 10: INPUT "VUOI SAPERNE L'AREA (Y/N
      )";A$
220  IF A$ = "Y" THEN  GOTO 250
230  GOTO 320
240  REM ****** CALCOLO DELL'AREA DEL POLIGONO *******
      *
250  FOR I = 2 TO COUNT - 1
260 A1 = A1 + (XCO(I - 1) * YCO(I))
270 A2 = A2 + (YCO(I - 1) * XCO(I))
280  NEXT I
290 A3 = XCO(COUNT - 1) * YFIRST:A4 = YCO(COUNT - 1) *
      XFIRST
300 AREA =  ABS ((A1 + A3 - A2 - A4) / 2)
310  VTAB 20: PRINT "L'AREA DEL POLIGONO E' = ";AREA
320  VTAB 22: HTAB 10: INPUT "VUOI VEDERNE LA IMMAGINE
       (Y/N)";A$
330  IF A$ = "Y" THEN  GOTO 500
340  HOME : VTAB 15: INPUT "VUOI ESAMINARE UN ALTRO PO
      LIGONO (Y/N) ";A$
350  IF A$ = "Y" THEN  GOTO 10
360  HOME : END
500  REM  ****** DISPLAY DEL DISEGNO SU APPLE 2 IN HGR
      *********
520  HGR2
525  HCOLOR= 3
530  FOR I = 1 TO COUNT - 1
540  HPLOT XCO(I),YCO(I) TO XCO(I + 1),YCO(I + 1)
550  NEXT I
560  REM  ***** DOPO CHE E' APPARSA LA FIGURA PREMERE
      RETURN PER RITORNARE IN TEXT MODE ********
570  INPUT A$: IF A$ = "" THEN  GOTO 600
600  TEXT : GOTO 340
1000  REM  ******CONTROLLO DELLE COORDINATE******
1010  REM  ******QUALORA SI VOLESSE IL DISEGNO DEL POL
      IGONO SULLO SCHERMO DISPONENDO DI UN APPLE 2 LE C
      OORDINATE NON DOVREBBERO SUPERARE I VALORI PERMES
      SI DALL'APPLESOFT*****
1020  REM   ******SI POTREBBE PERO' APPLICARE UN FATTO
      RE DI SCALA E ALLORA LE COSE CAMBIEREBBERO !!!!!!
      !!!!!
1030  IF XCO(COUNT) > 279 OR XCO(COUNT) < 0 THEN  GOTO
      1050
1040  IF YCO(COUNT) > 191 OR YCO(COUNT) < 0 THEN  GOTO
      1050
1045  GOTO 1060
1050 FLAG = 1
1060  RETURN
```

Figura 8

Figura 9 - Calcolo dell'area di un rettangolo mediante l'espressione:
$$1/2 \mid X_1 Y_2 + X_2 Y_3 + \dots + X_n Y_1 - Y_1 X_2 - Y_2 X_3 - \dots - Y_n X_1 \mid$$
Per verificare l'espressione data, facciamo il semplice caso di un rettangolo. In questo caso n = 4, per cui si ha:

$$\begin{aligned} \text{area} = & \mid 1/2 (X_1 Y_2 + X_2 Y_3 + X_3 Y_4 + X_4 Y_1) + \\ & - 1/2 (Y_1 X_2 + Y_2 X_3 + Y_3 X_4 + Y_4 X_1) \mid = \\ & = \mid 1/2 [(X_1 (Y_2 - Y_4) + X_2 (Y_3 - Y_1) + \\ & + X_3 (Y_4 - Y_2) + X_4 (Y_1 - Y_3)] \mid = \\ & = \mid 1/2 [(X_1 - X_3) (Y_2 - Y_4) + (X_2 - X_4) (Y_3 - Y_1)] \mid \end{aligned}$$

Dato che, a meno del segno, si ha
$$\mid X_1 - X_3 \mid = \mid X_2 - X_4 \mid = a$$
e
$$\mid Y_2 - Y_4 \mid = \mid Y_3 - Y_1 \mid = b$$
In definitiva si ottiene:
$$= 1/2 (ab + ab) = ab$$

a questo articolo di introduzione alla periferica digitizer una tabella rappresentantiva della situazione di mercato dei digitizers in Italia (v. Tabella I).
Le informazioni fornite nella tabella non devono essere considerate esaustive, ma indicative delle caratteristiche di ciascuna delle periferiche presentate.
È ovvio che il potenziale cliente di un digitizer, anche se un hobbista, ha particolari esigenze che potrà e dovrà discutere con il potenziale fornitore.
È poi da notare che le informazioni riportate si riferiscono a digitizer interfacciabili e/o interfacciati con personal e micro computers; a tal proposito è opportuno ricordare che la filosofia di sistemi grafici vuole che il digitizer sia servito da un micro o mini, ma spesso non è efficace e redditizio pensare di processare i dati provenienti da un digitizer tramite un micro: se non altro per la loro mole. ■

Bibliografia

1) Davis, M.R., T.O. Ellis, *The RAND tablet: a man machine Graphical Communication Device* (Proc. AFIPS FJCC 1964, 2355).

2) Roberts, L.G., *The Lincoln Wand* (Proc. AFIPS FJCC 1966, 223).

3) Science Accessiories Corp., *Graf/pen Sonic Digitizer* (Southport, Connecticut, 1970).

4) Teixeira, J.F., R.P. Sallen, *The Sylvania Tablet: A new approach to graphic data input* (Proc. AFIPS SJCC 1968, 315).

Tabella 1. Situazione di mercato di digitizer in Italia.

Nome prodotto	Interfaccia	Dimensioni (A)	Risoluzione	Trasmission mode	Tipo di cursore	Costo in lire	A chi rivolgersi in Italia
Apple (TM) graphics input tablet	Apple interface	28 x 28 cm. circa	200 linee per pollice	modo continuo e per punti	stilo	1.055.000	IRET Via Emilia Santostefano, 32 Reggio Emilia
Calcomp (C)	RS 232/C single & (B) dual port parallela GPIB	28 x 28 cm. circa altre dimens.	200 e 400 linee per pollice	incrementale puntuale continuo remoto	stilo puntatore puntatore con lente e tastiera	da 3 a 15 milioni	Calcomp Farnese, 1 Milano; Via Thailandia, 27 Roma
HI-PAD	RS 232/C parallela	28 x 28 cm. circa	200 linee per pollice	continuo puntuale	stilo puntatore	925.000	DB-Electronic Instruments Via Teano, 2 Milano
Bit PAD ONE (TM) summagraphics Corp (C)	RS 232/C parallela GPIB	28 x 28 cm. circa	200 linee per pollice	puntuale continuo	stilo puntatore puntatore lente e tastiera	1,7 milioni	Telecom srl Via Matteo Civitali, 75 Milano
Benson digitizer (C)	RS 232/C parallela	60 x 42 cm.	1000 linee per pollice	puntuale continuo	stilo puntatore (con visualizzatore numerico delle coordinate)	8,2 milioni	Benson S.p.A. Via Rivoltana, 8 Segrate (MI)

(A) – Le dimensioni sono quelle del piano di lavoro, cioè della superficie utile.

(B) – Permette di utilizzare il digitizer completamente staccato dal sistema ospitante.

(C) – I prezzi e le indicazioni si riferiscono ai digitizers di minima dimensione che vengono prodotti. Le case caratterizzate da questa notazione sono orientate verso prodotti di dimensione e prestazione maggiori.

Un periferico per lo svolgimento di calcoli numerici basato sul processore MM 57109

di **S. Valcher, C. Pagura**
Laboratorio di Polarografia ed Elettrochimica Preparativa del C.N.R., Padova

Si descrive un periferico numerico basato sul processore MM 57109 della National, dotato di una propria memoria da 512 x 4 bit (pari a 32 registri), interfacciato con una porta di I/O 8255 ad un microprocessore 8080.

Introduzione

In moltissimi casi, usando i microprocessori in apparecchi di misura o di controllo, ci si imbatte nel grosso problema di elaborare dei dati mediante funzioni più complesse di addizioni o moltiplicazioni quali le funzioni trascendenti o la estrazione di radice quadrata.
Varie sono le soluzioni possibili, sia nel costo che nell'efficienza e, comunque, indissolubilmente legate alle contingenze del particolare problema. Un fattore, certamente determinante nelle scelte proponibili, è la velocità con la quale si desidera ottenere un risultato con un dato numero di cifre significative sulle quali si deve operare: per esempio routines molto veloci fanno largo uso di tabulazioni delle funzioni di interessi.
Sia che vogliamo vedere aumentata la velocità, sia che vogliamo una maggiore precisione, la strada da percorrere è, a senso unico, verso un sempre maggior impegno di memoria.
Se il fattore *tempo di elaborazione* non è essenziale molti problemi vengono risolti; ma rimane sempre oneroso gestire e coordinare delle routines matematiche, soprattutto per chi lavora con diverse famiglie di microprocessori e deve forzatamente costruire una bibblioteca di programmi per ciascuna famiglia: dove un algoritmo si rivela efficiente per una certa architettura può non esserlo più con un'altra.
È evidente che questi oneri vanno accuratamente valutati prima di accingersi a spendere un tempo non trascurabile nell'implementare un software matematico, convenendo prendere in considerazione la possibilità di sostituire tutto il software particolare con un processore numerico affiancato all'hardware del sistema. Un vantaggio appare immediato: è una soluzione del tutto indipendente dalla famiglia di microprocessori se si eccettua, naturalmente, l'interffacciamento elettrico e di programmazione, invero riducibili a poche centinaia di bytes.
Altri vantaggi vengono di conseguenza: l'eliminazione del software matematico porta a una diminuzione di memoria ROM e RAM necessaria e la CPU liberata dall'esecuzione di questo software, è disponibile per altre operazioni che possono essere contemporanee all'elaborazione svolta dal processore numerico, cosa, quest'ultima, molto importante nel controllo di processi.
Citiamo, come esempio in cui l'impiego del processore numerico (organizzato in modo da essere più propriamente denominabile come *periferico numerico*) è risolutore di parecchi problemi, la nostra esperienza nell'automazione di misure elettrochimiche, in particolare nelle titolazioni automatiche, dove abbiamo, oltre alla necessità di controllare delle periferiche specializzate e dei convertitori A-D, una serie di pesanti linearizzazioni sui dati acquisiti (aventi andamenti esponenziali) con l'aggravante di dover lavorare su un campo di valori molto esteso e con un congruo numero di cifre significative. Se a questo punto si aggiunge la necessità di operare un "filtraggio" sull'insieme dei dati raccolti per trarne la massima informazione, ci si può rendere conto di quale mole di lavoro sarebbe stato lo sviluppo della sola parte di elaborazione, tenuto conto che l'obiettivo finale deve essere un apparecchio compatto e di costo ragionevole.
Inoltre, lavorare con 8 cifre decimali di mantissa e 2 cifre di esponente più i segni equivale a lavorare in binario su 28 bit di mantissa e 8 bit di esponente.

L'hardware

Sulla base di queste considerazioni abbiamo sviluppato un *periferico numerico* scegliendo come processore numerico l'MM57109 della National (già descritto nel numero 1 di BIT in un articolo di Peter Nelson[1] e dal quale abbiamo tratto una parte dell'hardware: col termine *periferico numerico* si intende un'organizzazione hardware intorno al chip 57109 che consente di facilitare il dialogo dello stesso con il sistema ospite e rende possibili prestazioni maggiori di quelle previste per il solo chip, per esempio un *allargamento* del suo set di istruzioni.
Lo schema di questo hardware è riprodotto in *figura 1*. A causa della diversa struttura dei dati numerici elaborati da un microprocessore e dal chip matematico, i processi di trasferimento di dati tra una unità e l'altra risultano piuttosto laboriosi e, quindi, ci è sembrato opportuno

[1] *Peter Nelson,* MM 57109: Un microprocessore per applicazioni numeriche *(Bit n. 1, Dicembre 1979).*

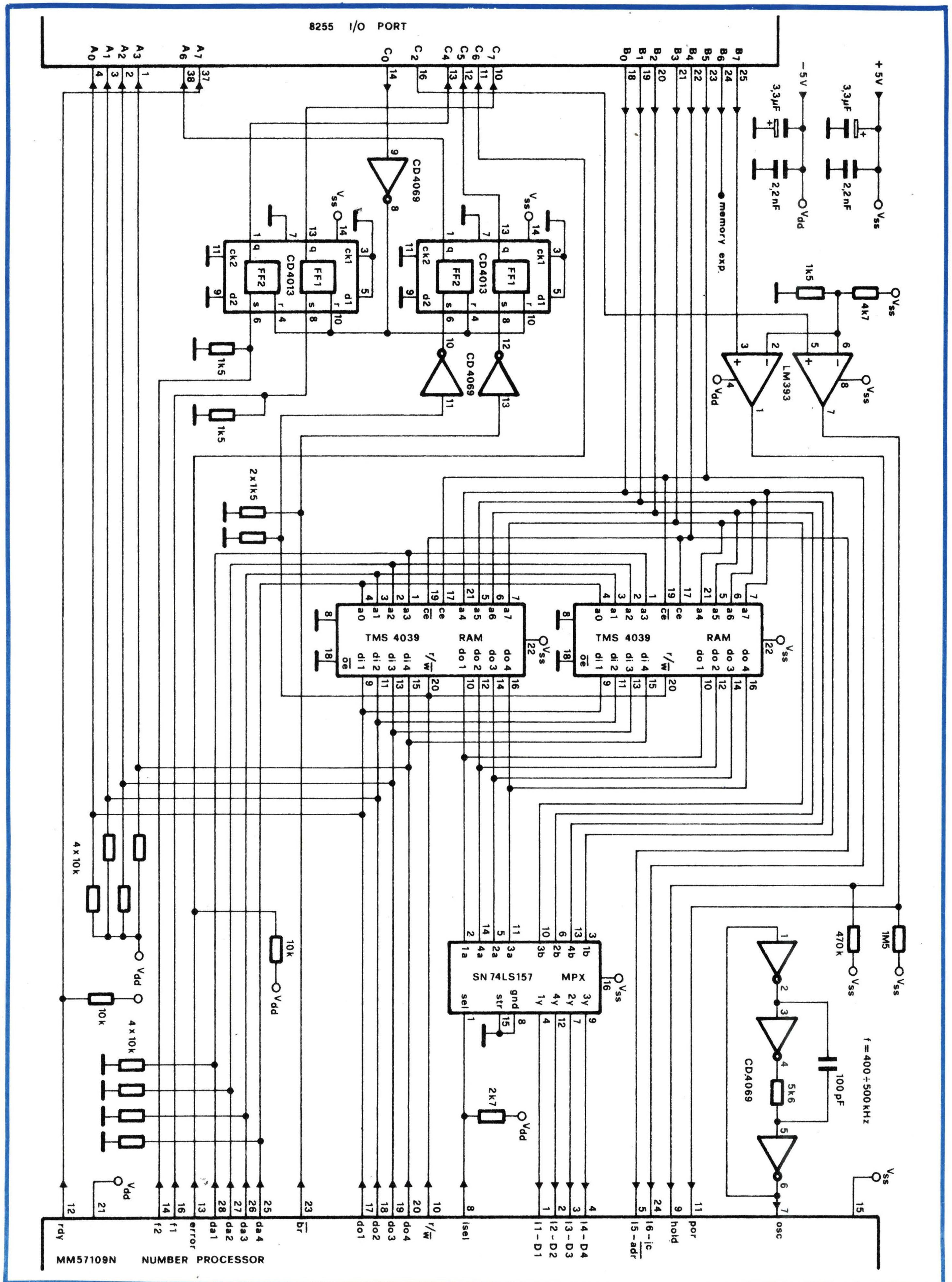



Figura 1 - Schema elettrico dell'interfacciamento dell'MM 57109 con un microprocessore 8080 tramite il chip di I/O 8255.

Un periferico

provvedere l'unità periferica di una sua memoria dati per ridurre la necessità di trasferimento ai soli dati finali.

Questa memoria è costituita dai 2 chip Texas TMS 4039 (1024 bit RAM organizzati 256 x 4) per un totale di 512 locazioni, che possono ospitare, nel formato compatibile con il 57109, 32 *registri* da 16 digit.

L'indirizzamento della memoria per i dati numerici avviene in parte tramite le linee uscenti dalla porta B dell'8255 e in parte tramite le linee DA fornite dal 57109: più precisamente, dall'esterno del periferico viene fornito l'indirizzo del *registro*, mentre, dall'interno, viene indirizzato il singolo digit, tra i 16 che formano il *registro*, con le 4 linee DA1÷DA4.

L'indirizzamento dall'esterno consta di 4 linee, che vanno ai bit 4÷7 dell'indirizzamento delle 4039, e 2 linee, che selezionano direttamente, tramite i chip enable, uno dei due chip: decodificando queste due linee ed eventualmente la linea B (a tale scopo lasciata libera), è facile ottenere un'espansione della memoria fino a 8 chip 4939, pari a 2048 x 4 bit e corrispondenti a 128 *registri*.

La logica di lettura/scrittura della memoria è governata direttamente dalla linea R/W del 57109.

Oltre a fornire l'indirizzamento dei registri, la porta B trasmette le istruzioni al processore.

Le istruzioni del 57109 sono costituite da gruppi di 6 bit e vanno presentate sulle linee $I_1 \div I_6$; nel nostro caso, 2 arrivano direttamente dalle linee B4 e B5 e 4 tramite il multiplexer 74157, che è pilotato dalla linea ISEL (*Instruction select*).

Quando questa linea è a livello logico alto le linee che fanno capo ai pin 1, 2, 3, 4, 5, 24 del 57109 sono abilitate a far entrare istruzioni, e quindi il multiplexer è commutato sulle linee B0÷B3. Quando ISEL cade al livello logico basso, come conseguenza di un'istruzione di input, le linee 1, 2, 3, 4 sono pronte a ricevere serialmente i 16 digit di un dato numerico, che vengono scanditi dalle linee DA1÷DA4, e il multiplexer commuta sulle linee di uscita delle memorie. Infine, la linea B7, sempre della porta B, è dedicata alla gestione della logica della linea HOLD, che sarà discussa in seguito.

L'integrato LM 393 ha il compito di fare da interfaccia elettrica ai livelli logici, diversi dal TTL, con cui lavorano le linee HOLD e POR (*Power on Reset*). La resistenza da 1,5 MΩ che collega il POR alla V_{ss} (+ 5 V) va cercata caso per caso (anche ∞) per ottenere un funzionamento affidabile.

I 4 flip-flop CD 4013 hanno il compito di trasformare i segnali a carattere impulsivo R/$\overline{W}$, $\overline{BR}$, F1 ed F2 (ottenibili con le istruzioni PF1 e PF2 del 57109), in livelli leggibili dalla porta C assieme alla linea ERROR; 4 linee della porta C sono programmate in uscita e comandano il POR e un comando di reset dei FF.

La porta A, programmata come input, legge la linea RDY (*Ready*), che il 57109 usa per indicare di essere pronto a ricevere un'istruzione, la linea R/$\overline{W}$ e le 4 linee D01÷D04.

In risposta ad una istruzione di OUT le D01÷D04 presentano, serialmente, i digit contenuti nel registro X del processore numerico; quest'operazione è sincronizzata dalla linea R/$\overline{W}$ che, passando dal livello logico alto a quello basso, indica che un digit valido è presente sulle linee D01÷D04.

Il software

Il periferico numerico, così come è organizzato, è in grado di accettare una istruzione, fornita alla porta B dal microprocessore e di eseguirla. D'altro canto, se l'istruzione da eseguire è un OUT, il microprocessore deve accollarsi il compito di leggere i digit che, serialmente, gli vengono forniti sulla porta A dell'8255.

La cosa più importante che deve fare il software che si occupa di gestire queste operazioni è quella di tener conto della logica delle linee RDY e HOLD. La linea RDY che, come già accennato, ci avvisa che il 57109 è pronto a ricevere ed eseguire una istruzione, e la linea HOLD, che permette di mettere in *wait* il 57109, fanno sì che si possa lavorare in modo asincrono. Essendo i tempi di esecuzione del microprocessore molto più brevi di quelli richiesti dal processore numerico, il lavorare in modo asincrono ci consente di ridurre il tempo di colloquio tra i due al minimo indispensabile, permettendo così al microprocessore (nel nostro caso un 8080) di svolgere altri compiti, mentre il numerico manda in esecuzione l'istruzione ricevuta.

Questo viene ottenuto con una routine (denominata SEND) che segue il diagramma di flusso della *figura 2*.

Nella prima fase, acquisendo ripetutamente la porta A, il microprocessore attende che il 57109 dia il segnale RDY; quindi presenta l'istruzione alla porta B ponendo nel contempo la linea HOLD uguale a 0 in modo tale che il 57109 la acquisisca. Nella terza fase il microprocessore attende che il segnale RDY vada basso, indicando che il 57109 sta acquisendo l'istruzione appena fornita; infine la linea HOLD viene posta a 1 affinchè il processore numerico, terminata l'esecuzione dell'istruzione corrente, si fermi in attesa della successiva.

Con queste modalità, il microprocessore resta a disposizione del processore numerico solo per la durata dell'impulso RDY, dopo di che il primo tra i due sistemi che esaurisce i suoi compiti si pone in attesa che l'altro sia disponibile per riprendere l'interazione.

Per il processo inverso, ovvero per il trasferimento di dati al microprocessore la routine può seguire la logica di quella apparsa nel già citato articolo di BIT.

Più interessante è l'ampliamento di queste funzioni, ottenibili lavorando su sequenze di istruzioni.

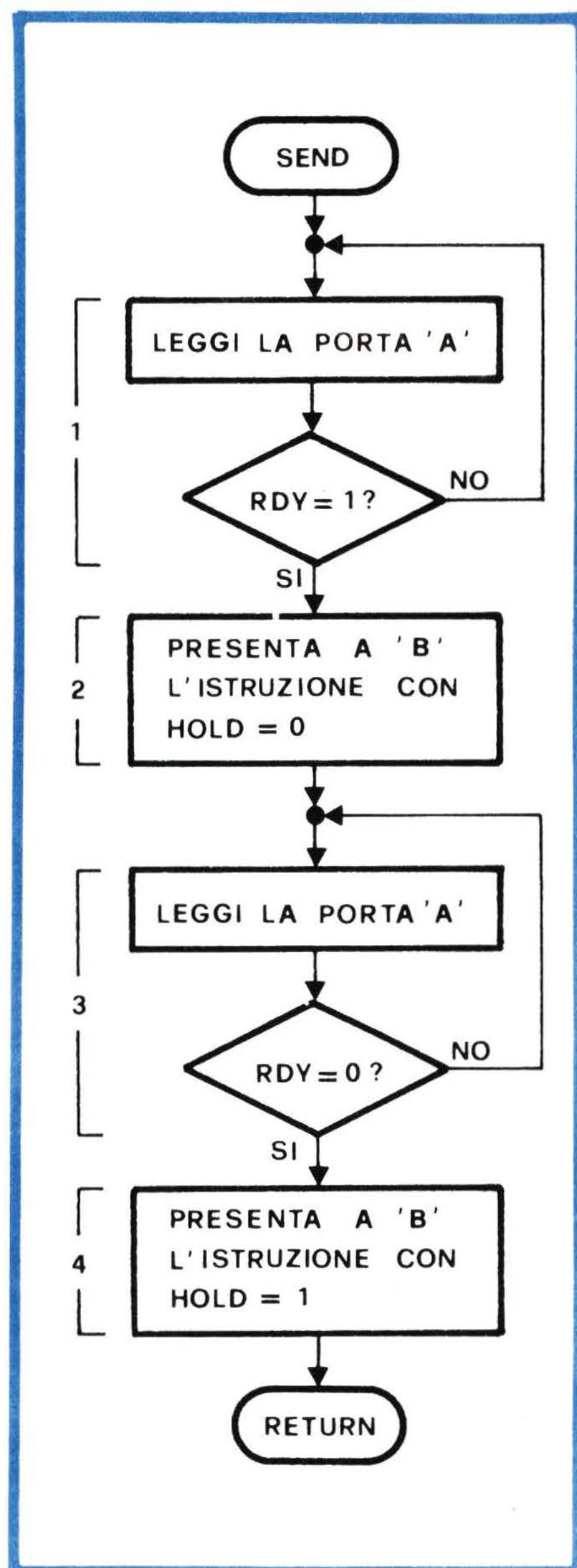



Figura 2 - Sequenza di invio di una istruzione (e dati) da 8080 (tramite 8255) al processore numerico MM 57109.

Un periferico

Il driver di I/O

Possiamo pensare di avere, invece che un semplice software di I/O che colloquia con il 57109, un software in grado di accettare delle *pseudoistruzioni* simili a quelle del 57109, di decodificarle e svolgere i vari compiti ad esse inerenti e coordinare l'invio delle vere istruzioni al 57109.
L'utente di questo *driver*, illustrato nello schema a blocchi della *figura 3*, non vede più il set di istruzioni proprio del processore numerico, ma un set un pò più vasto, che si basa sulla maggiore lunghezza della parola gestita dal microprocessore (8 bit) rispetto a quella delle istruzioni del numerico (6 bit), ma, soprattutto, sulla possibilità di una interpretazione preventiva delle pseudoistruzioni che può implicare una possibilità di scelte *indipendenti* dalla logica del processore numerico.
Questo tipo di prestazioni, nell'ambito della sezione del software in discussioni, interessa essenzialmente la funzione di output.
Il 57109 prevede l'istruzione di OUT avente il codice Hex 16 e la funzione di presentare serialmente sulle linee di output il contenuto del registro X.

Se al driver viene fornita la successione di parole 16-0X, dove X va da 0 a 9, questa esercita un controllo sul trasferimento del dato numerico ad un buffer nella memoria del microprocessore e/o sulla stampa dello stesso dato. In base al valore della seconda parola, è possibile discriminare il numero di cifre significative da stampare dopo la virgola, se la stampa deve essere preceduta o meno da un CR-LF o se si deve solo effettuare un trasferimento senza stampa. Oltre a ciò, viene effettuato un controllo sulla notazione con cui il dato è stato trasferito (se floating-point o scientifica) in modo da ottenere sempre un formato di stampa corretto.
In *figura 4* viene dato il listato di questa parte del driver. L'ingresso avviene tramite la WRTX? dopo che è stato riconosciuto un codice compreso tra 00 e 0A; questa routine va a vedere se il codice precedente era un 16 Hex (OUT), e in caso positivo passa il controllo alla routine di trasferimento. Può sembrare curioso che il riconoscimento della sequenza 16-0X avvenga sulla seconda parola, invece che sulla prima che parrebbe inequivocabilmente distinguere il significato della seconda; in realtà questo modo di procedere tiene conto della possibilità di gestione da tastiera del driver e permette di semplificare l'intervento del microprocessore nel caso di trasferimenti alle memorie del 57109.
La distinzione tra notazione scientifica e floating-point avviene nella routine XTYPE sulla base della presenza della parola 0B nel 4° byte nella stringa trasferita che caratterizza il primo tipo di notazione.
Il driver, nel suo complesso, è sufficiente (assieme ad opportune inizializzazioni) affinchè nello sviluppo di programmi per l'8080 si possono fare eseguire brevi sequenze di istruzioni o istruzioni singole al periferico numerico.

Figura 3 - Driver di I/O: a questo livello si è in grado di lavorare con un set di istruzioni più vasto di quello proprio del processore numerico, ed in un certo senso indipendente dal processore numerico stesso.

Procedura di esecuzione di un programma numerico

Se si rende necessario far svolgere al periferico un vero e proprio programma di elaborazione numerica, il driver deve essere controllato da un software opportuno, che gestisca un *contatore di programma* per il driver stesso.
In altre parole, occorre una procedura che, dato un programma costituito da una serie di istruzioni *valide per il driver* (pseudo-istruzioni) e risiedene nella memoria del sistema 8080, fornisca in sequenza queste istruzioni, tenendo conto dell'occorrenza o meno di salti (eventualmente condizionati) che modificano il corso del programma.
La procedura creata a questo scopo è denominata GORUN, ed opera, nel caso di pseudo-istruzione di salto, calcolando l'indirizzo della prossima pseudo-istruzione valida sulla base di uno shift relativo (generalmente entro il range di ± 128 passi di programma) dell'indirizzo dell'istruzione corrente, indicato dal 2° byte della pseudo-istruzione di JUMP (*v. Tabella I*). La pseudo-istruzione di salto (eventualmente condizionato) può essere trasformata in una pseudo-istruzione di chiamata di subroutine, previa attivazione del flag 2 tramite la pseudo-istruzione CF (corrispondente alla PF2 del 57109). All'interno della subroutine, la pseudo-istruzione Set Return Flag (mnemonico RF corrispondente a PF1) posta immediatamente prima dell'ultima pseudo-istruzione, segnala il termine della stessa e fa ritornare il controllo alla routine chiamante.
In altre parole, i segnali F_2 e F_1 del 57109 sono utilizzati per l'esecuzione di chiamate di su-

Figura 4 - Listato di alcune routines del driver di I/O. La routine SEND segue a grandi linee il diagramma di flusso di Figura 2 e presenta alcuni controlli predisposti per il resto del s/w. Chiamando la routine SEND, con il registro A contenente l'istruzione pronta, viene effettuato il semplice invio, mentre la chiamata di INSTRS (normalmente fatta dalla procedura GORUN) prima di effettuare l'invio controlla l'occorrenza o meno di una stampa del registro X. La routine DESEND cura l'invio di 4 cifre BCD, risultanti ad esempio da un'acquisizione A-D, giacenti in DE.

```
                                DRIVER LISTING                    PAGE  1
LOC   OBJ CODE    STMT SOURCE STATEMENT                           ASM 1.1

                     1 ;
                     2 ; INVIO AL 57109 DI QUATTRO CIFRE BCD CONTENUTE IN DE
0000  CD0400         3 DESEND CALL DSEND    ;INVIA PRIME DUE CIFRE
0003  53             4        LD D,E        ;SPOSTA IN D SECONDE DUE CIFRE
0004  7A             5 DSEND  LD A,D        ;PORTA IN A CIFRE
0005  0F             6        RRCA
0006  0F             7        RRCA          ;E SPOSTA IN 4 BIT LOW
0007  0F             8        RRCA          ;CIFRA 4 BIT HIGH
0008  0F             9        RRCA
0009  CD0D00        10        CALL ASEND    ;E INVIA AL 57109
000C  7A            11        LD A,D        ;PORTA IN A CIFRE
000D  E60F          12 ASEND  AND 0FH       ;ISOLA CIFRA LOW
000F  C31B00        13        JP SEND       ;INVIO ULTIMA CIFRA E RITORNO
                    14 ;
                    15 ; INVIO CODICE ISTR. A 57109 O TRASFER. E STAMPA REG. X
0012  FE0A          16 INSTRS CP 0AH        ;ISTR. DI TRASFERIMENTO ?
0014  D21B00        17        JP NC,SEND    ;SE "NO", INVIA
0017  CD4F00        18        CALL WRTX?    ;OPPURE ALTRA ISTR. ?
001A  C8            19        RET Z         ;RITORNO AVENDO ESEGUITO TRASF.
                    20 ;
                    21 ; ATTESA RDY, INVIO ISTRUZIONE AL 57109 ED AVVIO
                    22 ; ESECUZIONE PREDISPONENDO HOLD AL PROSSIMO RDY
001B  E5            23 SEND   PUSH HL       ;SALVA PUNTATORE
001C  D5            24        PUSH DE
001D  2101FF        25        LD HL,OUTPRT  ;CARICA INDIRIZZO PORTA OUT
0020  47            26        LD B,A        ;SALVA ISTR. IN B
0021  E63F          27        AND 3FH       ;ELIMINA HOLD E POR DA ISTR.
0023  57            28        LD D,A        ;E SALVALA IN D
0024  F640          29        OR 40H        ;AGGIUNGI HOLD ALL'ISTR.
0026  5F            30        LD E,A        ;E SALVALA IN E
0027  79            31        LD A,C        ;PORTA IN A ISTR. PRECED.
0028  48            32        LD C,B        ;ISTR. CORRENTE ANCHE IN C
0029  FE10          33        CP 10H
002B  DA3500        34        JP C,HWAIT    ;NON ISTR. BRANCH
002E  FE1B          35        CP 1BH
0030  D23500        36        JP NC,HWAIT   ;NON ISTR. BRANCH
                    37                      ;SE PRECED. ISTR.= BRANCH,
0033  0E3F          38        LD C,3FH      ;ISTR. "NO-OP" IN C
0035  3A00FF        39 HWAIT  LD A,(INPRT)  ;CARICA IN A PORTA RDY
0038  17            40        RLA           ;PORTA RDY NEL CARRY
0039  D23500        41        JP NC,HWAIT   ;ATTENDI RDY
003C  72            42        LD (HL),D     ;PRESENTA ISTR. PRIVA DI HOLD
003D  3A00FF        43 LWAIT  LD A,(INPRT)  ;CARICA IN A PORTA RDY
0040  17            44        RLA           ;E PORTA RDY NEL CARRY
0041  DA3D00        45        JP C,LWAIT    ;ATTENDI /RDY
0044  73            46        LD (HL),E     ;PRESENTA ISTR. + HOLD
0045  B3            47        OR E          ;PONI /Z MEDIANTE E≠0
0046  D1            48        POP DE
0047  E1            49        POP HL        ;RICARICA PUNTATORE
0048  C9            50        RET
```

```
                                DRIVER LISTING                    PAGE  2
LOC   OBJ CODE    STMT SOURCE STATEMENT                           ASM 1.1

                    51 *EJECT
                    52 ;
                    53 ; COMANDO STAMPA REGISTRO X
0049  3E16          54 WRITEX LD A,16H      ;ISTR. "W"
004B  CD1B00        55        CALL SEND     ;INVIA
004E  AF            56        XOR A         ;0=ISTR. TRASF. A 8080 E STAMPA
                    57 ;
                    58 ; DISCRIMINAZIONE TRA ISTR. TRASF. E INDIRIZZO
004F  D5            59 WRTX?  PUSH DE
0050  57            60        LD D,A        ;SALVA ISTR. CORRENTE
0051  79            61        LD A,C        ;CARICA PRECEDENTE ISTR.
0052  FE16          62        CP 16H        ;CONFRONTA CON ISTR. "W"
0054  7A            63        LD A,D        ;RICARICA ISTR. CORRENTE
0055  CC5A00        64        CALL Z,WRTX   ;ESEGUI TRASF. X SE RICHIESTO
0058  D1            65        POP DE
0059  C9            66        RET
                    67 ;
                    68 ; TRASFERIMENTO DA REG. X A BUFFER 8080 CON OPZIONI STAMPA
005A  F5            69 WRTX   PUSH AF       ;SALVA FLAG PER IL RITORNO
005B  E5            70        PUSH HL       ;SALVA PUNTATORE PROGRAMMA
005C  2AE02F        71        LD HL,(OUTVEC);CARICA INDIRIZO BUFFER
005F  E5            72        PUSH HL       ;E SALVALO NELLO STACK
0060  CD1B00        73        CALL SEND     ;FALSA ISTR. DI INDIRIZZAMENTO
                    74 ;
                    75 ; ESECUZIONE TRASFERIMENTO
0063  CDEC00        76 TRANSX CALL FLRES    ;RESET FF DI FLAG
0066  3A00FF        77 TSTRDY LD A,(INPRT)  ;CARICA IN A BIT DI FLAG E DIGIT
0069  17            78        RLA           ;PORTA RDY NEL CARRY
006A  DA7800        79        JP C,TRXEND   ;SE RDY=1 TRASFER. FINITO
006D  17            80        RLA           ;PORTA WR NEL CARRY
006E  D26600        81        JP NC,TSTRDY  ;SE WR=0 ATTENDI
0071  1F            82        RRA           ;SE BIT DIGIT VALIDI
0072  1F            83        RRA           ;RIAGGIUSTA INPUT
0073  77            84        LD (HL),A     ;E MEMORIZZALO
0074  23            85        INC HL        ;INCREM. PUNTAT. BUFFER
0075  C36300        86        JP TRANSX     ;CONTINUA TRASFERIMENTO
                    87 ;
0078  D1            88 TRXEND POP DE        ;RICARICA INDIRIZZO BUFFER
0079  78            89        LD A,B        ;RICARICA IN A ISTR.
007A  0F            90        RRCA          ;BIT COMANDO 'CR-LF' NEL CY
007B  4F            91        LD C,A        ;E SALVA IN C IL RESTO
007C  D40010        92        CALL NC,TCRLF ;STAMPA SU NUOVA LINEA
007F  79            93        LD A,C        ;RESTO ISTR. IN A
0080  E604          94        AND 4         ;BIT FLAG /STAMPA
0082  CC8800        95        CALL Z,XTYPE  ;STAMPA NUMERO
0085  E1            96        POP HL        ;RICARICA PUNTATORE PROGR.
0086  F1            97        POP AF        ;RICARICA FLAG E ISTR.
0087  C9            98        RET
                    99 ;
                   100 ; STAMPA CONTENUTO BUFFER IN NOTAZ. SCIENT. O FL. POINT
                   101 ; CIFRE DECIMALI = 0,1,2 O TUTTE
0088  7D           102 XTYPE  LD A,L        ;PORTA IN A PARTE LOW PUNTAT. READX
0089  93           103        SUB E         ;CALCOLA IL NUMERO DI CIFRE
008A  C8           104        RET Z         ;SE CONTATORE = 0 RITORNO (ERRORE)
008B  D5           105        PUSH DE       ;SALVA TOP DEL BUFFER
008C  13           106        INC DE        ;SALTA CIFRE ESPONENTE
008D  13           107        INC DE        ;E SEGNI IPOTIZZANDO
008E  13           108        INC DE        ;NOTAZ. SCIENT.
```

```
                                DRIVER LISTING                    PAGE  3
LOC   OBJ CODE    STMT SOURCE STATEMENT                           ASM 1.1

008F  2602         109        LD H,02       ;FLAG DI NOTAZ. SCIENT. IN H
0091  D604         110        SUB 4         ;E PONI CONTATORE - 4
0093  6F           111        LD L,A        ;NEL REG. L
0094  DA9F00       112        JP C,FLTPNT   ;SE DIGIT TRASFER. < 4
0097  1A           113        LD A,(DE)     ;CARICA POS. PUNTO N.SC.
0098  E60F         114        AND 0FH       ;ISOLA BIT SIGNIFICATIVI
009A  FE0B         115        CP 0BH        ;0BH SE N.SC., DIGIT BCD SE FL.P.
009C  CAA500       116        JP Z,SIGN     ;SE NOTAZ. SCIENT.
009F  2600         117 FLTPNT LD H,00       ;FLAG FL. POINT
00A1  1B           118        DEC DE        ;SPOSTA PUNTAT. SU
00A2  1B           119        DEC DE        ;POS. PUNTO DEC. FL.P.
00A3  2C           120        INC L
00A4  2C           121        INC L         ;E AGG. CONTATORE
00A5  1B           122 SIGN   DEC DE        ;PASSO INDIETRO
00A6  1A           123        LD A,(DE)     ;CARICA SEGNI
00A7  13           124        INC DE        ;PUNTAT. A PUNTO DEC.
00A8  B4           125        OR H          ;AGGIUNGI FLAG NOTAZ.
00A9  67           126        LD H,A        ;E SALVALI IN H
00AA  E608         127        AND 08H       ;ISOLA SEGNO MANTISSA
00AC  C40011       128        CALL NZ,TMINUS;SE 1XXX STAMPA '-'
00AF  1A           129        LD A,(DE)     ;CARICA POS. PUNTO DEC.
00B0  13           130        INC DE        ;PUNTAT. A PRIMA CIFRA MANT.
00B1  2F           131        CPL           ;TRASFORMA POSIZ.
00B2  E60F         132        AND 0FH       ;PUNTO DECIMALE
00B4  D603         133        SUB 3         ;IN CONTATORE
00B6  47           134        LD B,A        ;E PONI IN B
00B7  CDE700       135 MANTDG CALL TDENUM   ;STAMPA CIFRA MANT. E AGG. PUNTAT.
00BA  2D           136        DEC L         ;AGG. CONTATORE
00BB  CAD500       137        JP Z,EXP      ;VAI AD ESPON. SE CIFRE FINITE
00BE  05           138        DEC B         ;DECREM. CONTAT. POS. PUNTO
00BF  C2B700       139        JP NZ,MANTDG  ;CONTINUA STAMPA CIFRE
00C2  79           140        LD A,C        ;RESTO ISTR. IN A
00C3  E603         141        AND 3         ;FLAG NUMERO COMPLETO ?
00C5  CACD00       142        JP Z,TPOINT   ;SE "SI", STAMPA PUNTO E CONT.
00C8  3D           143        DEC A         ;NUMERO CIFRE DEC.
00C9  CAD500       144        JP Z,EXP      ;SE 0, PASSA A ESPONENTE
00CC  6F           145        LD L,A        ;DEC. NEL CONTATORE
00CD  3E2E         146 TPOINT LD A,'.'
00CF  CD0012       147        CALL TYPE     ;STAMPA PUNTO
00D2  C3B700       148        JP MANTDG     ;CONTINUA STAMPA
00D5  7C           149 EXP    LD A,H        ;CARICA SEGNI E FLAG
00D6  E603         150        AND 03H       ;ISOLA SEGNO ESP. E FLAG
00D8  D1           151        POP DE        ;RICARICA TOP DEL BUFFER
00D9  C8           152        RET Z         ;SE ZERO FLOT. POINT
00DA  3E5C         153        LD A,'\'      ;STAMPA SIMBOLO
00DC  CD0012       154        CALL TYPE     ;SEPARAZIONE ESPONENTE
00DF  7C           155        LD A,H        ;RICARICA SEGNI
00E0  1F           156        RRA           ;PORTA SEGNO ESPONENTE IN CY
00E1  DC0011       157        CALL C,TMINUS ;SE XXX1 STAMPA '-'
00E4  CDE700       158        CALL TDENUM   ;STAMPA LE
00E7  1A           159 TDENUM LD A,(DE)     ;DUE CIFRE
00E8  13           160        INC DE        ;DELL' ESPONENTE
00E9  C30013       161        JP TALOW
```

```
                                DRIVER LISTING                    PAGE  4
LOC   OBJ CODE    STMT SOURCE STATEMENT                           ASM 1.1

                   162 *EJECT
                   163 ;
                   164 ; RESET DEI FLAG
00EC  E5           165 FLRES  PUSH HL
00ED  2103FF       166        LD HL,CNTPRT
00F0  3600         167        LD (HL),0     ;RESET CØ
00F2  3601         168        LD (HL),1     ;SET CØ
00F4  E1           169        POP HL
00F5  C9           170        RET
                   171 ;
                   172 INPRT  EQU  0FF00H
                   173 OUTPRT EQU  0FF01H   ;I/O MEMORY MAPPED
                   174 CNTPRT EQU  0FF03H
                   175 ;
                   176 OUTVEC DEFL 2FE0H    ;CONTIENE INDIR. BUFFER DI TRASFER.
                   177 ;
                   178 ; SUBROUTINE DI STAMPA UTILIZZATE
                   179 TCRLF  DEFL 1000H    ;'CR,LF' A TTY
                   180 TMINUS DEFL 1100H    ;'-' A TTY
                   181 TYPE   DEFL 1200H    ;CONTEN. DI A A TTY
                   182 TALOW  DEFL 1300H    ;CONVERSIONE IN ASCII-HEX
                   183                      ;DI 4 LSB DI A E STAMPA

  0 ASSEMBLY ERRORS
```

broutines e dei relativi ritorni.
Le possibilità di ampliamento del set di pseudo-istruzioni già vista per il driver può assumere importanza ancora maggiore nel caso di procedure a livello più elevato.
Nell'ambito della routine GO-RUN la chiave di tale ampliamento è rappresentata dal codice FF Hex (mnemonico RE) che ha la funzione di segnalare una modificazione del flusso del programma.
Nel caso più semplice esso indica la fine di un programma di elaborazione numerica, ma, usato assieme alla rivelazione del flag F1 e F2 e del flag di branch, assume significati più complessi.
Per esempio, la sequenza JMP, RE 1X, YY permette di effettuare un salto al di fuori del range ± 128 passi, (raggiungibile dagli 8 bit tramite la sequenza JMP, YY): quando essa è riconosciuta, grazie alla presenza di RE, fa effettuare un salto alla locazione calcolata mediante uno spostamento relativo "X" di pagina e uno spostamento relativo "YY" di locazione. Anche in questo caso l'attivazione (tramite la pseudo-istruzione CF) del flag di Call trasforma l'istruzione di salto in una chiamata di subroutine con le modalità già dette per il ritorno.
L'uso della pseudo-istruzione speciale RE in una sequenza di pseudo-istruzione può essere ampliato ulteriormente: un'altra funzione utile è quella che permette di chiamare un programma 8080 (come un'acquisizione dati o una stampa di messaggi) direttamente dall'interno di un programma numerico. Ad esempio la sequenza CF, NOP, RE, XX fa eseguire la procedura 8080 il cui indirizzo è in una tabella predisposta e puntata da XX: eseguito il programma 8080, il controllo ritorna al programma del numerico. Il vantaggio di tale funzione risiede nel poter scrivere una procedura unica (un programma per il numerico) includendo anche operazioni 8080 senza dover tener due amministrazioni separate, una per le procedure numeriche, e una per quelle 8080.
La logica seguita dalla procedura GORUN è mostrata nel diagramma di flusso di *Figura 5*. L'esecuzione viene fatta chiamando la procedura GORUN, avendo nei registri HL il puntatore predisposto alla prima istruzione del programma da eseguire; al termine del programma, indicato da una istruzione RE, il controllo ritorna alla procedura che ha chiamato la GORUN.

Compilazione di un programma numerico

Assieme alle procedure di GO-RUN e del driver di I/O, per evitare di sviluppare i programmi di elaborazione numerica scri-

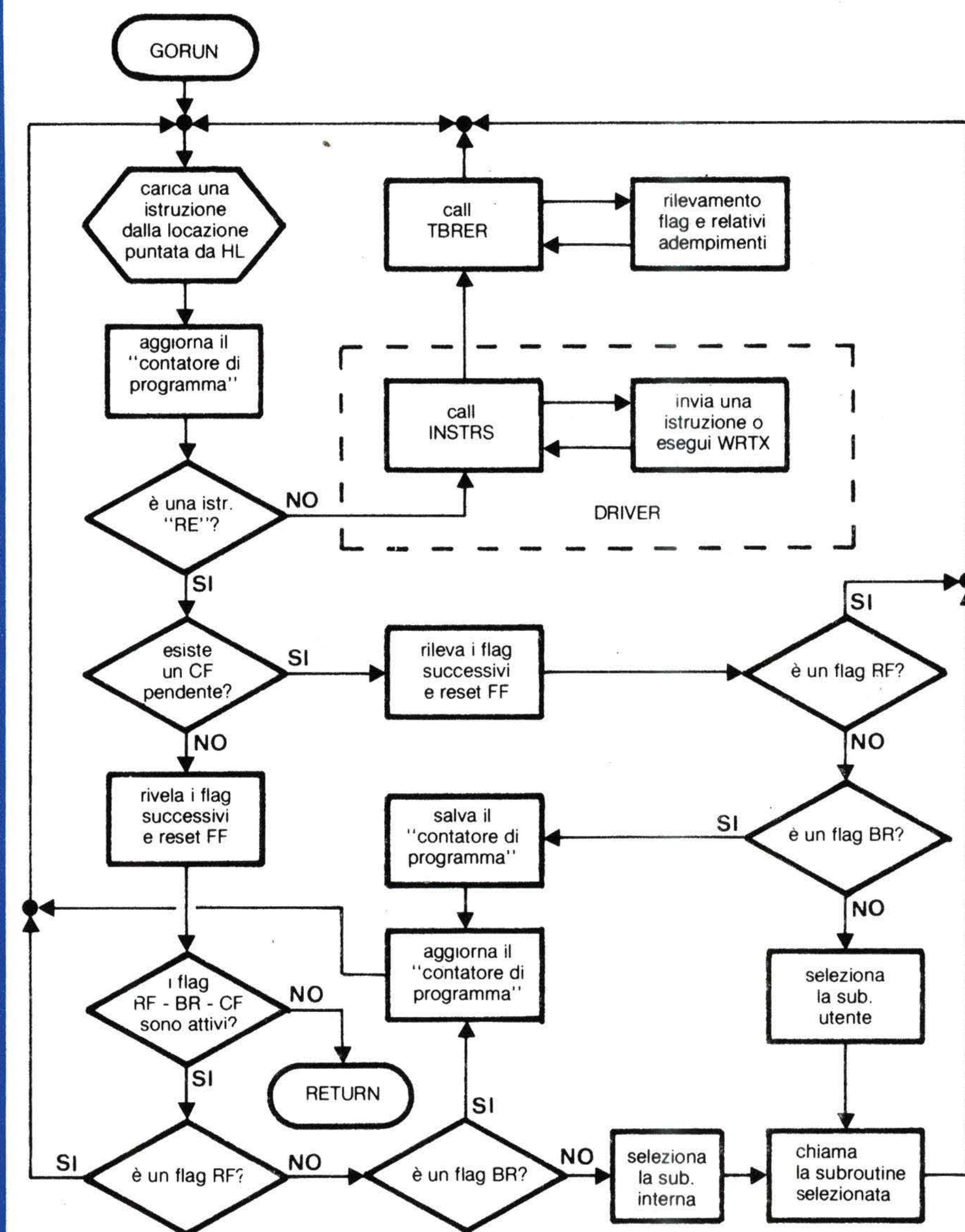



Figura 5 - Diagramma di flusso della procedura GORUN di esecuzione di un programma "numerico".

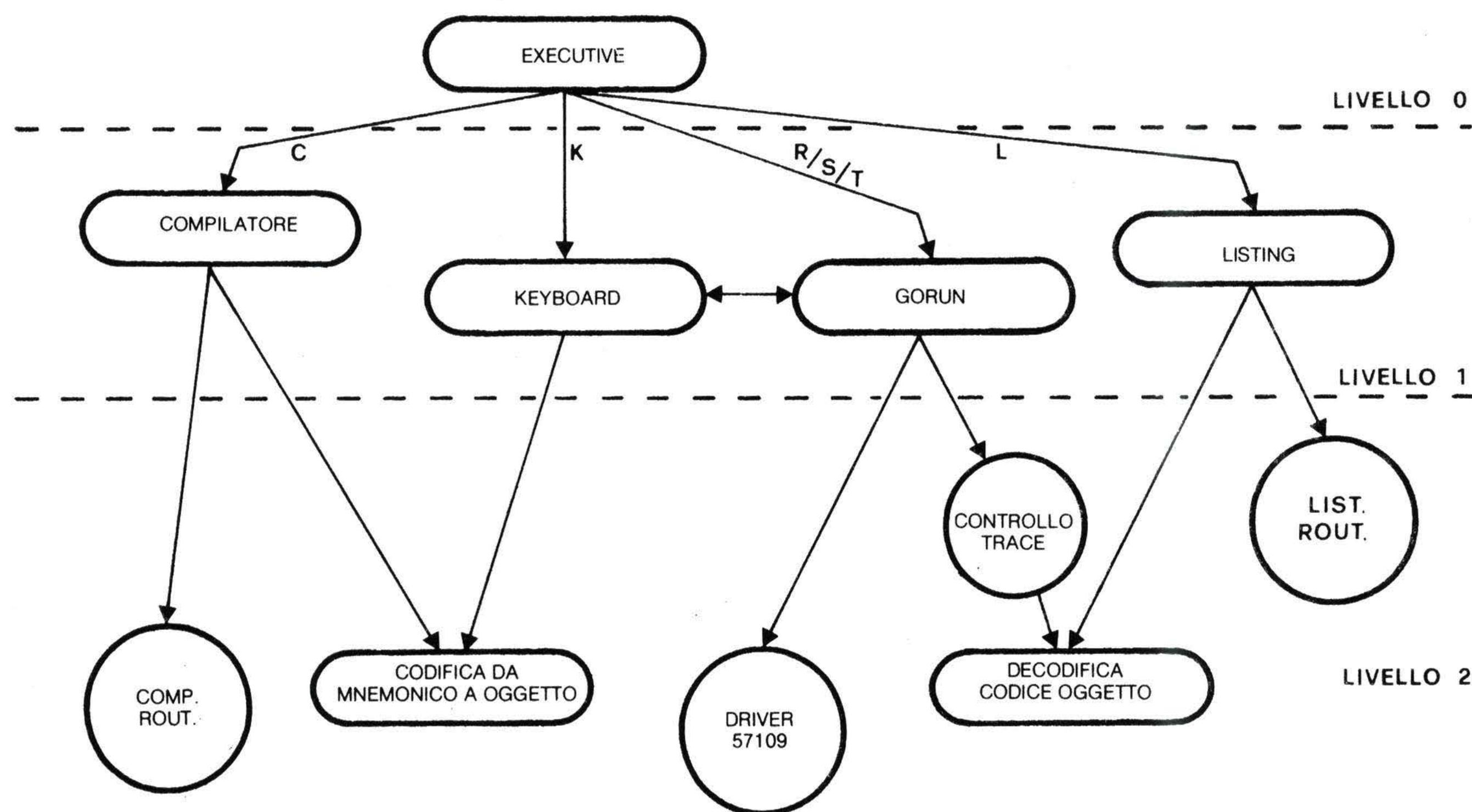



Figura 6 - Il grafo evidenzia la struttura logica del package NUMEX. Il nucleo EXECUTIVE riconosce i comandi C, K, R, S, T, L e chiama uno dei sottoprogrammi del primo livello di "nest" rispetto al nucleo.

```
: C 2400
2400: FF: p          0D    ;x<-3.1415927
2401: FF: re         FF    ;introduci il raggio: x<-R, y<-3.1415927
2402: FF: sq         33    ;x<-R*R
2403: FF: *          3B    ;x<-R*R*3.1415927
2404: FF: w          16    ;stampa x
2405: FF;p           00    ;a capo
2406: FF: j          15    ;salta
2407: FF:00a         F8    ;a 2400
2408: FF: re         FF    :fine
2409: FF: re         FF    :del programma
240A: FF: er               ;fine della compilazione

: L 2400
2400: 0D        p
2401: FF        re
2402: 33        sq
2403: 3B        *
2404: 16        w
2405: 00        p
2406: 15        j
2407: F8        2400
2408: FF        re
2409: FF
```

Esempio 1: Compilazione e listato di un programma per il calcolo dell' area di un cerchio.

```
: C 2400
2400: FF: #05 p      0D    ;il label 05 viene attribuito alla loc. 2400
2401: FF: re         FF
2402: FF: sq         33
2403: FF: *          3B
2404: FF: w          16
2405: FF:p           00
2406: FF: j          15    ;salta al label
2407: FF:#05         FE    ;     05
2408: FF: re         FF
2409: FF: re         FF
240A: FF: er

: L 2400
2400: 0D        p
2401: FF        re
2402: 33        sq
2403: 3B        *
2404: 16        w
2405: 00        p
2406: 15        j
2407: FE        2400
2408: FF        re
2409: FF
:
: L 2400
2400: 0D        p
2401: FF        re
2402: 33        sq
2403: 3B        *
2404: 16        w
2405: 00        p
2406: 15        j
2407: F8        2400       che appare nei listing successivi
2408: FF        re
2409: FF
:
```

Esempio 2: Lo stesso programma dell' Es.1, ma compilato con l'uso dei label.

```
: C 2400
2400: FF: re         FF    da tastiera viene dato il numero dei cerchi X<-N
2401: FF: st         1C    M<-X (N)
2402: FF: #01 re     FF    si passa il controllo alla tastiera per l' input del raggio
2403: FF: cf         2A    se il raggio e` negativo chiama la sub. 1
2404: FF: tn         12
2405: FF:#02         FE
2406: FF: rf         28    reset (in questo caso non indipensabile)
2407: FF: re         FF    della logica del flag di chiamata
2408: FF: cf         2A    chiama la sub. 2 che calcola l' area del cerchio
2409: FF: j          15
240A: FF:#03         FE
240B: FF: dm         1A    decrementa il contatore (M<-M-1) e salta a 01
240C: FF:#01         FE    se M non e` zero
240D: FF: re         FF    altrimenti termina l' esecuzione
240E: FF: re         FF
240f: FF: #02 _      0C    SUB1      cambia segno, X<-X-1
2410: FF: w          16    stampa X
2411: FF:q           01    senza 'CR,LF'
2412: FF: rf         28    ritorna al programma chiamante
2413: FF:            3F    'SP'=NOP
2014: FF: #03 sq     33    SUB2      X=r*r
2415: FF: p          0D    X<-π, y<-r*r
2416: FF: *          3B    X<-r*r*π
2417: FF: w          16    stampa X
2418: FF:p           00    con 'CR,LF'
2419: FF: rf         28    ritorna al programma chiamante
241A: FF:            3F    'SP'+NOP
241B: FF: re         FF    segnale di fine programma
241C: FF: re         FF
241D: FF: er               fine compilazione

: L 2400                        : L 240F
2400: FF        re              240F: 0C        _
2401: 1C        st              2410: 16        w
2402: FF        re              2411: 01        q
2403: 2A        cf              2412: 28        rf
2404: 12        tn              2413: 33        sq
2405: FE        240F            2414: 0D        p
2406: 28        rf              2415: 3B        *
2407: FF        re              2416: 16        w
2408: 2a        cf              2417: 00        p
2409: 15        j               2418: 28        rf
240A: FE        2413            2419: 3F
240B: 1A        dm              241A: FF        re
240C: FE        2402            241B: FF
240D: FF        re              :
240E: FF
:
```

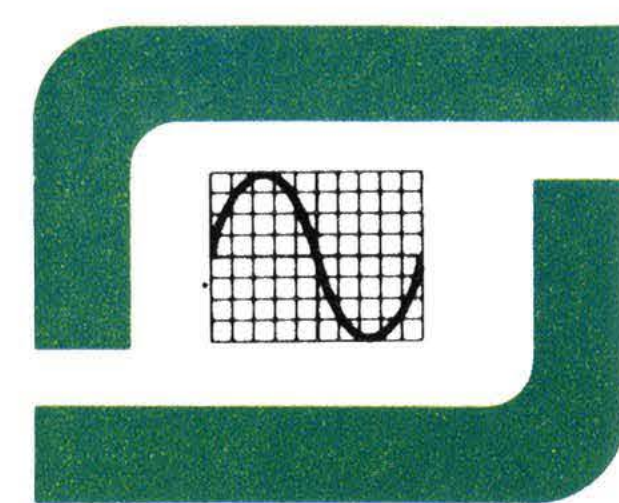
Esempio 3: Uso delle chiamate di subroutine in un programma dimostrativo che calcola l' area di N cerchi; se il raggio dato e` negativo esso viene stampato dopo cambio di segno.

CLASSE	SOTTOCLASSE	MNEMONICO	CODICE	DESCRIZIONE
CONTR.		re ('CR')	FF	SCAMBIO CONTR. PROGR.-TAST.
INPUT NUMER.		0	00	CIFRE PER IMPOSTAZIONE NUMERICA
		1	01	
		2	02	
		3	03	
		4	04	
		5	05	
		6	06	
		7	07	
		8	08	
		9	09	
		. [DP]	0A	PUNTO DECIMALE
		\ (l) [EE]	0B	IMPOST. ESPONENTE
		_ [CS]	0C	CAMBIA SEGNO ALL' ESPONENTE O ALLA MANTISSA
		p [PI]	0D	PI GRECO IN X
SPOST.	STACK	up [EN]	21	SHIFT VERSO L'ALTO DELLO STACK
		dw [POP]	2E	" " IL BASSO DELLO ST. - DOWN
		rl [ROLL]	23	ROTAZIONE VERSO IL BASSO - ROLL
		xy [XEY]	30	SCAMBIO X CON Y
		xm [XEM]	1B	SCAMBIO X CON M
		st [MS]	1C	X VA IN M - STORE
		rc [MR]	1D	M VA IN X - RECALL
	REG. X	sl [LSH]	1E	SHIFT VERSO SINISTRA DELLA MANT.
		sr [LSH]	1F	" " DESTRA " "
MATEM.	F(X,Y)	+	39	X=Y+X
		-	3A	X=Y-X
		* [x]	3B	X=YxX
		/	3C	X=Y:X
		y^ (y^) [YX]	38	X=Y**X - Y ALLA X
	F(X,M)	m + [M+]	20,39	M=M+X
		m - [M-]	20,3A	M=M-X
		m * [MX]	20,3B	M=MxX
		m / [M/]	20,3C	M=M:X
	F(X)	rp [1/X]	37	X=1:X - RECIPROCO
		rt [SQRT]	34	X=X**1/2 - RADICE DI X
		sq [SQ]	33	X=X**2 - QUADRATO DI X
		d^ (d^) [10X]	32	X=10**X - DIECI ALLA X
		e^ (e^) [EX]	31	X=e**X - EXP(X)
		ln [LN]	35	X=LOGARITMO NAT. DI X
		lg [LOG]	36	X=LOGARITMO DECIM. DI X
		si [SIN]	24	X=SEN(X)
		co [COS]	25	X=COS(X)
		ta [TAN]	26	X=TAN(X)
		a si [SIN^{-1}]	20,24	X=ARC SEN(X)
		a co [COS^{-1}]	20,25	X=ARC COS(X)
		a ta [TAN^{-1}]	20,26	X=ARC TAN(X)
		dr [DTR]	2D	CONV. GRADI-RADIANTI DI X
		rd [RDT]	2C	" RADIANTI-GRADI DI X
TERMIN.		[NOP]	3F	NO-OPERATION,FINE INPUT - BLANCK
BRANCH	JUMP	j NN [JMP]	15,NN	SALTO INCONDIZIONATO
	COND. BR.	t0 NN [TX=0]	11,NN	" SE X=0
		t1 NN [TXF]	13,NN	" SE |X|<1
		tn NN [TXLT0]	12,NN	" SE X NEGATIVO
		te NN [TERR]	14,NN	" SE ERROR FLAG
	CONTATORE	im NN [IBNZ]	19,NN	INCR. MEMORIA E SALTA SE M=0
		dm NN [DBNZ]	1A,NN	DECR. " ", " " M=0
LONG BRANCH	JUMP	j FFr tw...j,NN	15,FF, 1x,NN	SALTO INCONDIZIONATO CON INCR. O DECR. INDIR. HIGH tw=10H->P-3, t0=11H->P-2, tn=12H->P-1 t1=13H->P+1, te=14H->P+2, j=15H->P+3 P=pagina
SUBR.	CALL	cf [PF2]	2A	FLAG INDICA CHE SUCCESSIVO BRANCH DEVE INTENDERSI CALL SUBROUTINE
	RETURN	rf [PF1]	28	FLAG DI RITORNO DA SUBROUTINE
	RES. FLAG	rf re	28,FF	RESET CALL FLAG PER CALL SUBR. COND.
SUBR. 8080		cf re N	2A, FF,N	CHIAMATA DA 57109 DI SUBR. 8080 O MISTA DA TAV. SISTEMA
		cf ISTR. re N	2A,XX, FF,N	CHIAMATA DA 57109 DI SUBR. 8080 O MISTA DA TAV. USER
I/O	MEMORIA	in [IN] 00...1F	17, 50..6F	MEM.EST. 00-1F VA IN X - INMem
		w [OUT] 00...1F	16, 50..6F	X VA IN MEM.EST. 00-1F - WRTMem
	8080A	w p...v	16, 00..09	X VA IN BUFFER 8080 CON OPZIONI CODICE OPZIONI: BIT Ø = /A CAPO - STESSA LINEA BIT 1,2 = N. DECIMALI + 1 SE Ø, INTERA MANTISSA BIT 3 = /STAMPA - SOLO TRASFER.
CLEAR		cl [MCLR]	2F	MASTER CLEAR
		ce [ECLR]	2B	ERROR FLAG = 0
OUT MODE		tg [TOGM]	22	TOGGLE DA NOTAZ. SCIENT. A FL. POINT E VICEVERSA
		sm [SMDC] 0...8	18, 00..08	DETERMINA NUM. DI CIFRE DA TRASFERIRE - SET MANT. n

(mn)=MNEMONICO UTILIZZABILE INDIFFERENTEMENTE
[MN]=MNEMONICO PROPOSTO DAL COSTRUTTORE DEL CHIP
Caratteri maiuscoli o minuscoli possono essere indifferentemente usati per i simboli rappresentati con minuscole.

Tabella I - Istruzioni: mnemonici e codici esadecimali.

Figura 7 - Alcuni esempi di "dialogo" fra utente e NUMEX nella fase di compilazione. I caratteri entro i riquadri sono introdotti dall'utente, mentre la seconda colonna mostra il contenuto corrente delle locazioni di memoria sia durante il listing che durante la compilazione.

Un periferico

EXEC.	EX. ROUT.	R <ADD>	INIZIO ESECUZIONE PROGR. - RUN
		T <ADD>	INIZIO ESECUZIONE PROGR. - TRACE
		S <ADD>	INIZIO ESECUZIONE PROGR. - SINGLE STEP
		K	CONTROLLO A TASTIERA
	PROGRAMM.	C <ADD>	COMPILAZIONE PROGR.
		L <ADD>	LISTING HEX - MNEMONICO, LABEL ASSEMB.
CONTR.		er FE	RITORNO AD EXECUTIVE

<ADD> = INDIRIZZO OPZIONALE PER DEFAULT L'EXECUTIVE MANDA AD INDIRIZZO FISSO

Tabella II - Comandi principali del package NUMEX.

vendo direttamente i codici operativi, si è reso necessario un software di supporto per la compilazione e il test dei programmi numerici.
Questo package, nel complesso denominato NUMEX, permette di:

1) Compilare i programmi con dei codici mnemonici consentono l'uso di label come riferimento per gli indirizzamenti e con la possibilità di ottenere dei listati comodi da consultare.
2) Permettere l'uso del periferico, tramite la TTY, come un normale calcolatore tascabile.
3) Fare eseguire i programmi compilati con la possibilità di procedere passo-passo e stampando i risultati intermedi (TRACE).

L'organizzazione dei vari programmi del NUMEX è mostrata nello schema a blocchi di *Figura 6*.
Dal nucleo principale, EXECUTIVE, mediante un'interpretazione di opportuni comandi si passa il controllo ai vari programmi; questi, a loro volta, organizzano una serie si subroutines di cui alcune sono comuni anche agli altri (come quelle del driver), mentre altre sono particolari (come quelle che danno il format nelle stampe di LISTING).

In *Figura 7* sono riportati alcuni esempi di "dialogo" tra utente e NUMEX nella fase di compilazione. I codici mnemonici sono stati scelti realizzando un compromesso tra l'uso del minor numero di caratteri alfanumerici, la comprensibilità dei codici stessi e le necessità relative all'interpretazione degli stessi da parte della routine Compiler, tenendo conto, dov'era possibile, del codice mnemonico originario del 57109.
L'elenco di questi mnemonici e la loro correlazione con quelli del 57109 è dato nella *Tabella I*, mentre i principali comandi del NUMEX sono dati nella *Tabella II*.
L'intero NUMEX nella sua versione più sofisticata occupa 2 kbytes di EPROM, escludendo le solite routine di monitor per il governo delle periferiche come TTY, video, etc. da esso chiamate. L'ingombro di RAM dipende solo dalla complessità dei programmi di elaborazione numerica, ma resta sempre molto contenuto. ■

Data Base personale

di **F. Del Vecchio, M. Valsasina**

Parte II

P I P P O CR P L U T O CR P A P E R I N O CR

Figura 1 - Esempio di file sequenziale.

BYTES LIBERI

C A N E CR 3 7 4 CR | U C C E L L O CR 2 4 5 CR

RECORD — RECORD

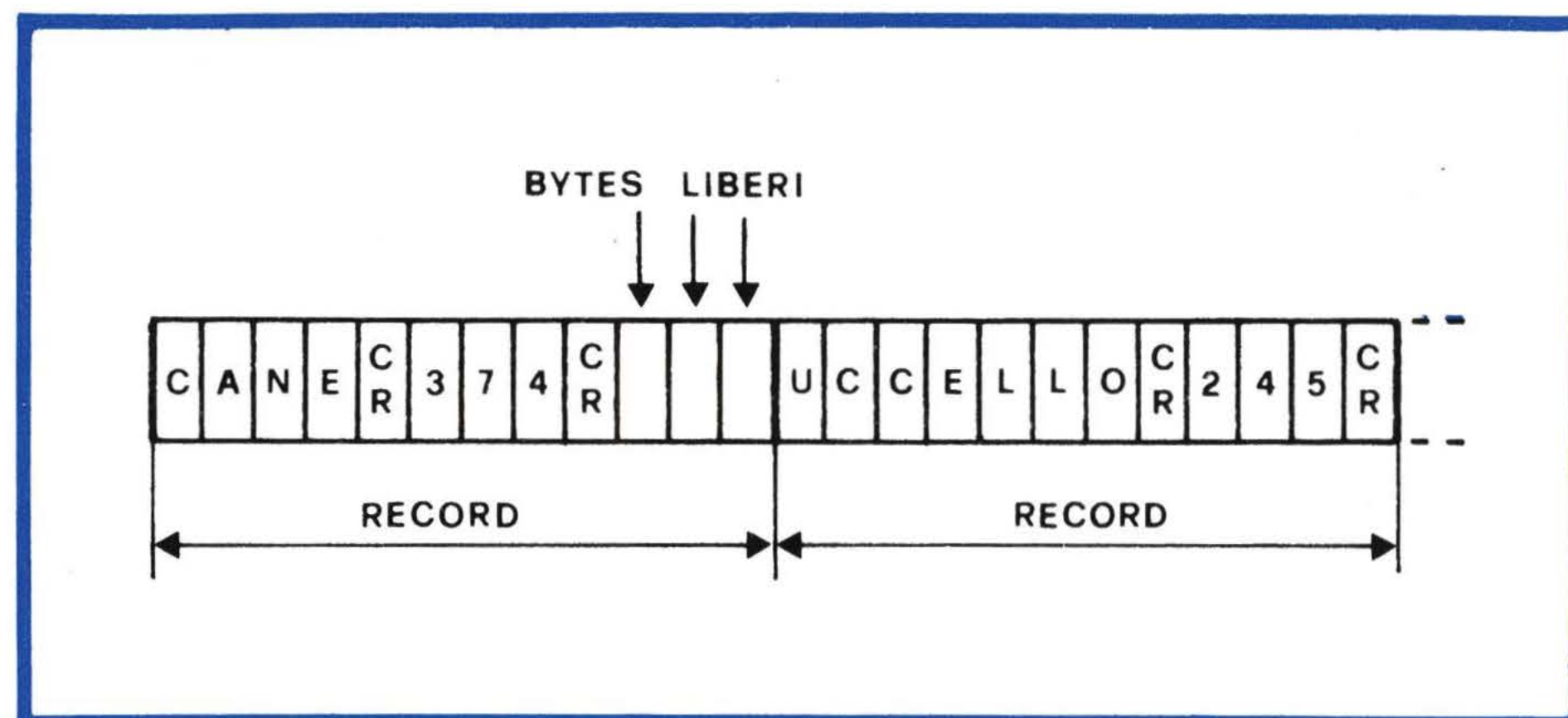



Figura 2 - Esempio di file random di lunghezza 12 bytes.

Acquisiti i criteri generali di struttura dati, descritti nella Parte I, approfondiamo lo studio di fattibilità, per definire l'architettura del Data Base intorno alla quale costruire il programma di gestione.
In questo articolo dovremmo necessariamente essere più pratici e fare i conti con gli strumenti hardware-software disponibili sul personal computer, nel caso specifico Apple II Plus. Dalla noiosa teoria passiamo quindi alla progettazione, certo più stimolante.

Il D.O.S. Apple

Oltre al linguaggio di programmazione Basic è necessario conoscere quali sono le funzionalità del sistema Apple nel trattamento delle informazioni sui mini floppy disk.
Il sistema in esame ha una struttura hardware flessibile ed espandibile, disponendo di 8 slot di uscita non dedicati al collegamento di specifiche periferiche, tranne lo slot 0, studiato per espansioni di linguaggio e lo slot 7, riservato alla scheda PAL per visualizzazione a colori. I sei slot rimanenti possono indifferentemente essere utilizzati per collegare stampanti, drive per mini floppy, ed altri tipi di periferiche. Ogni interfaccia per mini floppy può pilotare 2 drive. Per accedere alle informazioni residenti su un floppy è necessario, qualora possa verificarsi ambiguità, definire slot e drive di appartenenza. In un floppy inoltre possono coesistere informazioni relative a programmi memorizzati, aree di memoria in formato binario, e files di dati di tipo sequenziale o random, cioè con suddivisione in records a lunghezza fissa. Le modalità di trattamento sono diverse per ogni categoria: per i nostri scopi sarà sufficiente conoscere il trattamento riservato ai programmi Applesolft ed ai files di dati. Ogni gruppo di informazioni o files è individuato da un nome di 30 caratteri al massimo: si può così richiamare programmi o accedere ad archivi direttamente per nome.
La gestione dei programmi è particolarmente semplice: dopo aver scritto il programma nella memoria del sistema tramite la tastiera è possibile salvarlo scrivendo SAVE <nome >, caricarlo da disco nella memoria dell'Apple scrivendo LOAD <nome>, eseguirlo scrivendo RUN <nome > o semplicemente RUN se il programma è già in memoria.

I files di dati (text) si dividono in due categorie:
— *files sequenziali*, nei quali le singole informazioni sono disposte consecutivamente, separate da un carattere di fine campo CR (*Carriage Return*) (*v. Figura 1*). In tal caso, dovendo cambiare un dato con uno di lunghezza maggiore, è necessario riscrivere tutto il file. Per contro un file sequenziale ottimizza lo spazio su disco e può essere utile per memorizzare parametri fissi, definiti una volta ed aggiornati solo raramente.
— *files random*, nei quali l'insieme di dati è suddiviso in records di lunghezza fissa contenenti uno o più campi separati ancora da CR (*v. Figura 2*). Per aggiornare un dato è sufficiente riscrivere il relativo record; inoltre è possibile accedere direttamente ad un record per numero, senza dover scandire tutti i dati sequenzialmente.
Per il trattamento dei files dati individuiamo quattro funzioni essenziali:
— OPEN: permette di aprire un flusso informativo con un file residente fisicamente su un certo floppy ed inoltre, se la struttura del file è di tipo random, permette di definire anche la lunghezza del record.
Esempio:

Sequenziale OPEN < nome >,
S < n. slot >,
D < n. drive >
Random OPEN < nome >,
L < lun. rec >,
S < n. slot >,
D < n. drive >

— READ: permette di indicare da quale file verrà effettuata la prossima lettura di dati con lo statement BASIC INPUT. Nel file sequenziale la prima lettura (input) sarà relativa al primo dato e le successive input leggeranno i dati consecutivamente. In un file random il criterio è identico ma relativo al singolo record, cioè è necessario definire nella READ anche il numero del record sul quale effettuare la lettura.
Esempio:

Sequenziale READ < nome >
Random READ < nome >,
R < n. record >

— WRITE: permette di individuare su quale file effettuare la prossima scrittura di dati con lo statement BASIC PRINT. Anche in questo caso le scritture sono consecutive da inizio file o ennesimo record in dipendenza del tipo di file sequenziale o random.
Esempio:

Sequenziale WRITE
< nome >
Random WRITE
< nome >,
R < n. record >

— CLOSE: permette di chiudere un flusso informativo ed assicura la scrittura delle precedenti informazioni. In tal caso non vi sono differenze sintattiche tra files sequenziali e random.

Data Base personale

Esempio:

```
Sequenziale CLOSE
            < nome >
Random      CLOSE
            < nome >
```

Tutti i comandi di controllo flusso sopra elencati, per essere riconosciuti ed eseguiti dal DOS (sistema operativo gestione dischetti), devono comparire come argomento di uno statement PRINT preceduti dal carattere di controllo CONTROL-D, pari a CHR$ (4). Il DOS Apple permette molti altri comandi di controllo, ma riteniamo sufficienti i quattro descritti per la gestione del Data Base in questione; gli interessati all'argomento potranno trovare esaurienti spiegazioni nel manuale DOS 3.2 Apple. Concludiamo con un esempio completo di statement BASIC di controllo DOS:

```
10 PRINT CHR$ (4)
"WRITE PIPPO, R12"
```

Strutturiamo il Data Base

Come precedentemente descritto, l'obbiettivo del progetto consiste nell'organizzare un archivio di Base accessibile con una chiave preferenziale e (supponiamo) quattro chiavi accessorie, che possa essere agganciato ad un secondo archivio gestito a liste, logicamente correlate ai record Base. Esistono quindi in prima analisi i seguenti archivi logici:

— Un archivio Base con relativi indice principale (MASTER INDEX) e indici accessori.
— Un archivio a Lista completo dei dati necessari alla gestione. Soffermiamoci sull'archivio Base cercando di individuare come possa essere *fisicamente* strutturato.

Tutte le informazioni dell'archivio Base, cioè i dati relativi ai vari campi costituenti i vari records, possono essere raccolti in un file che chiameremo MASTER FILE (MST.FL) di tipo ovviamente random per facilitarne le operazioni di scrittura, lettura e modifica. La lunghezza del record Base sarà quindi la somma delle lunghezze dei relativi campi con relativo carattere di fine campo CR (v. *Figura 3*).

Per velocizzare accessi, inserimenti e cancellazioni del record Base non manterremo tale file ordinato, per cui accederemo ad esso usando un dizionario di *chiavi-puntatori* costituente un file MASTER INDEX (o MST.INX) che terremo invece sempre ordinato e presente sia in memoria sia su disco (v. *Figura 4*). I due files sono fisicamente separati anche se logicamente costituiscono un archivio unico perchè inscindibili, dovendo sempre accedere al dizionario per raggiungere ordinatamente le informazioni di Base. In tal modo è possibile solo l'accesso per chiave, cioè per il primo campo del record Base assunto come identificatore prioritario.

Nulla però vieta di costruire ulteriori dizionari chiave-puntatore, ove la chiave può essere uno qualsiasi nei campi del record

Figura 3 - Struttura random del MASTER FILE.

Figura 4 - Accesso col metodo del dizionario.

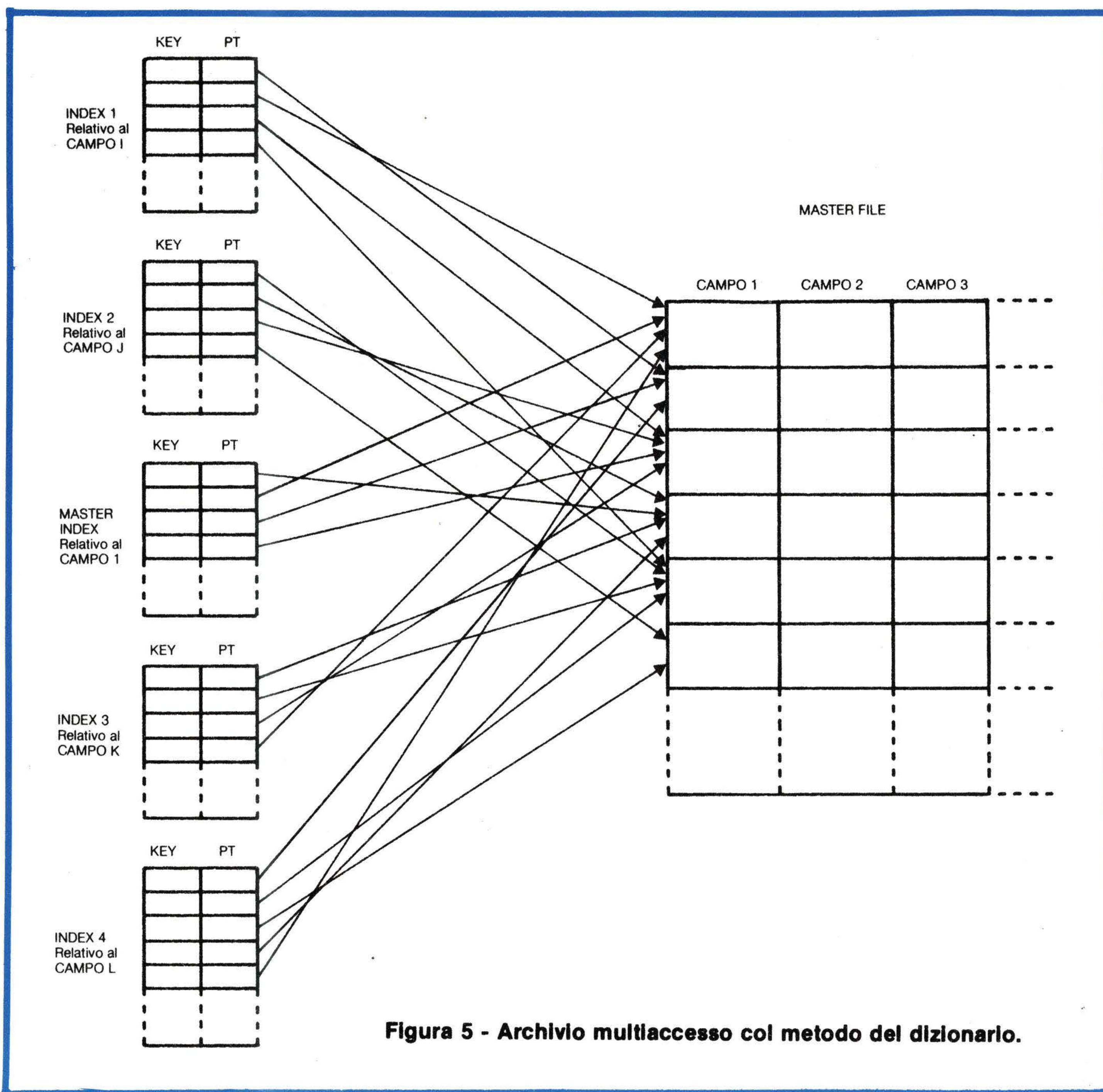



Figura 5 - Archivio multiaccesso col metodo del dizionario.

Base: pensiamo che quattro indici accessori possano essere sufficienti (*v. figura 5*).

Naturalmente in una struttura multiaccesso di questo tipo risultano complesse e lunghe le fasi di inserimento e cancellazione, dovendo non solo aggiornare il singolo record del MASTER FILE e riordinare il MASTER INDEX, ma effettuare analoga operazione su tutti i files indice accessori. Per velocizzare la procedura terremo sempre agiornato il MASTER INDEX e permetteremo l'aggiornamento degli indici accessori *solo su comando esplicito*.

La struttura dell'archivio Base comincia a prendere consistenza; manca ora un file accessorio ove memorizzare lo stato del Data Base, cioè:

— il numero di records validi NX;

— il numero di elementi allocati EA, cioè la somma degli elementi validi e dei cancellati. (L'inserimento prevede di occupare lo spazio reso disponibile dalle cancellazioni, e solo ad esaurimento di questo ultimo allocare nuovi elementi);

— la presenza o meno di un archivio gestito a liste, definita da un flag FL (flag-link);

— il tipo di stampante collegata PT (0 per parallela, 1 per seriale);

— il numero di colonne gestibili dalla stampante CN;

— il flag indicatore di disco pieno DF;

— il numero del campo associato al primo indice accessorio IA (1, 0);

— la lunghezza della chiave, (eventualmente troncata) relativa al primo indice accessorio IA (1. 1);

— la validità del primo indice accessorio IA (1, 2)

ed altre analoghe informazioni corrispondenti agli ultimi tre casi ma relative al secondo, terzo e quarto indice accessorio.

Sono quindi in totale 18 i parametri che permettono di staticizzare lo stato dell'archivio: questi saranno memorizzati in un file sequenziale che chiameremo DATA BASE CONTROL o brevemente DB. CTRL.

Riassumendo, per la gestione Bata Base avremo:
DB. CTRL: controllo Data Base, file sequenziale
MST. INX: dizionario prioritario, file random
MST. FL: archivio dati, file random
INDEX. 1: primo dizionario accessorio, file random
INDEX. 2: secondo dizionario accessorio, file random
INDEX. 3: terzo dizionario accessorio, file random
INDEX. 4: quarto dizionario accessorio, file random

Definita la struttura dell'archivio Base, consideriamo ora come poter agganciare una gestione a lista e come strutturarla per una facile e veloce gestione.

Dallo studio teorico precedentemente sviluppato si nota che una gestione a lista necessita di puntatori che leghino i records fra di loro e con la testa della lista. Nel nostro caso la testa della lista è logicamente ogni singolo record base per cui deriva la necessità di inserirvi un puntatore. Una soluzione di tale tipo vincola però ad allocare spazio nel MASTER FILE anche nelle applicazioni in cui si vuole rinunciare alla gestione a liste.

Si preferisce allora una soluzione altenativa, che permette inoltre di renderne più veloce la successiva gestione. Invece di tenere i puntatori alle liste all'interno dei records, possiamo pensare di estrarli e costruire un file random (che chiameremo LINK ENTRY o LNK. ENTRY), costituito quindi da un vettore i cui elementi sono le teste delle liste. Per verificare l'esistenza (nell'archivio Liste) di records correlati all'*i-esimo*

Data Base personale

record Base, basterà leggere l'*i-esimo* elemento del vettore LINK ENTRY: se questo è 0 non sono agganciati records, altrimenti il numero costituirà il puntatore al primo record della lista associata.
Una lista monodirezionale è costituita da coppie record-puntatore, ove quest'ultimo indica la successiva coppia correlata: tale accoppiamento logico anche in questa occasione preferiamo suddividerlo in un vettore di puntatori (file LINK POINTERS o LNK.PTR) ed un file random di record LINK FILE o LNK.FL, contenente le informazioni aggiuntive del record Base di appartenenza. Riassumiano quindi il concetto in *Figura 6*.
Non è una gestione molto semplice, lo ammettiamo, ma ritorneremo sull'argomento.
Manca ancora una serie di parametri che permettano di conoscere la struttura del record relativo al file LINK, cioè:
— il numero di campi, e per ognuno di essi
 a) nome del campo
 b) tipo del campo (1 alfanumerico, 2 numerico)
 c) lunghezza del campo

Questo insieme di parametri costituirà un file di controllo del LINK che chiameremo LNK. CTRL.

Riepilogando, l'archivio a liste sarà costituito dai seguenti files:

LNK.CTRL	: file sequenziale contenente i parametri
LNK.ENTRY	: file random, contenente il vettore di acces- so alle liste (teste)
LNK.PTR	: file random contenente il vettore dei poin- ters gestione liste
LNK.FL	: file random, contenente i record gestiti a li- ste

Abbiamo definito la struttura dell'archivio Base e dell'archivio gestito a liste: manca però ancora la definizione di alcuni parametri quali:

— il numero massimo di records base ammessi,
— il numero dei campi del record base e per ogni campo
 a) il nome del campo
 b) il tipo del campo (1 alfanumerico, 2 numerico)
 c) la lunghezza del campo

Inoltre potrà essere interessante poter parametrizzare anche la locazione ed organizzazione fisica dei files, memorizzando per ciascuno:

 a) il nome del file
 b) l'eventuale lunghezza del record (0 per sequenziale)
 c) lo slot di appartenenza
 d) il drive di appartenenza

costruendo poi automaticamente le relative stringhe di comando DOS.
Inseriremo tale insieme di parametri in un ulteriore file sequenziale SYSTEM CONTROL o SYST.CTRL. Questo file risiederà sempre nello slot 6, drive 1, e sarà *l'unico file fisso*, dal quale poter definire qualsiasi configurazione ed allocazione fisica del Data Base.
Per ottimizzare il programma definiamo per ogni comando OPEN, READ, WRITE, CLOSE dei vettori O$ (), R$ (), W$ (), C$ () in modo da eseguire il comando sul file desiderato direttamente, ad esempio: PRINT D$;O$ (I), ove I è l'indice indicante il file relativo, come da *Tabella I*. La parametrizzazione rende certo meno leggibile il listing, ma si ottiene una notevole versatilità e semplicità di gestione.

Il programma

Definita la struttura non resta che procedere top-down nell'analisi del programma che suddivideremo in due parti ben distinte:
— configurazione del sistema o SETUP
— gestione DATA BASE o DB

La prima permetterà di configurare il sistema, e cioè: definire il formato dell'archivio e nel caso del MASTER FILE indicare quanti campi ne costituiscono il record e quali sono i nomi, tipi e lunghezze ad essi associati. Si tratta di definire i parametri contenuti nel SYST.CTRL, non desiderando certo caricarli a mano.
Discorso analogo per il DB. CTRL e per il LNK.CTRL.

MASTER FILE — LINK ENTRY — LINK POINTERS — LINK FILE — EM MASSIMO NUMERO EL.

Figura 6 - Struttura dell'archivio gestito a liste.

SYST.CTRL	**Controllo dell'intero sistema di gestione (sequenziale)**
0 MST.INX	**Master index o dizionario principale (random)**
1 MST.FL	**Archivio base (random)**
2 DB. CTRL	**Controllo del Data Base (sequenziale)**
3 LNK.CTRL	**Controllo dell'archivio gestito a liste**
4 LNK.ENTRY	**Vettore di aggancio dell'archivio base alle liste (random)**
5 LNK.PTR	**Vettore dei puntatori per gestione delle liste (random)**
6 LNK.FL	**Archivio gestito a liste (random)**
7 INDEX.1	**Primo indice accessorio**
8 INDEX.2	**Secondo indice accessorio**
9 INDEX.3	**Terzo indice accessorio**
10 INDEX.4	**Quarto indice accessorio**

Tabella I - I files costituenti la struttura del Data Base.

Figura 7 - Data Base top.

1.1

INIZIALIZZA VARIABILI 6000

LETTURA FILE SYST. CTRL 1.1.3

LETTURA FILE DB. CTRL 1.1.4

SI — I FILES ESISTONO — NO

CREAZIONE DATA BASE 1.1.1 60060

MENÙ PRINCIPALE SETUP 1.1.2 60500

Figura 8 - SETUP.

1.1.1

ERRORE ≠ "END OF DATA" — SI — SEGNALA IL TIPO DI ERRORE

NO

DEFINIZIONE ALLOCAZIONE DATA BASE 1.1.1.1

DEFINIZIONE CAMPI DATA BASE 1.1.1.2

60095

SET PARAMETRI LUNGHEZZA RECORD

SCRITTURA FILE SYST. CTRL 1.1.1.3

LETTURA FILE SYST. CTRL 1.1.3

SCRITTURA FILE DB. CTRL 1.1.1.4

SCRITTURA DEI FILE MASTER 1.1.1.5

MODIFICHE — SI — CANCELLA I FILE MASTER

NO

SCRITTURA FILE SYST. CTRL 1.1.1.3

SCRITTURA FILE DB. CTRL 1.1.1.4

Figura 9 - Creazione del Data Base.

In tal modo è possibile *personalizzare* il Data Base che, a parte le funzionalità standard, deve poter archiviare indirizzi, cataloghi o tabelle senza dover toccare uno statement di programma.

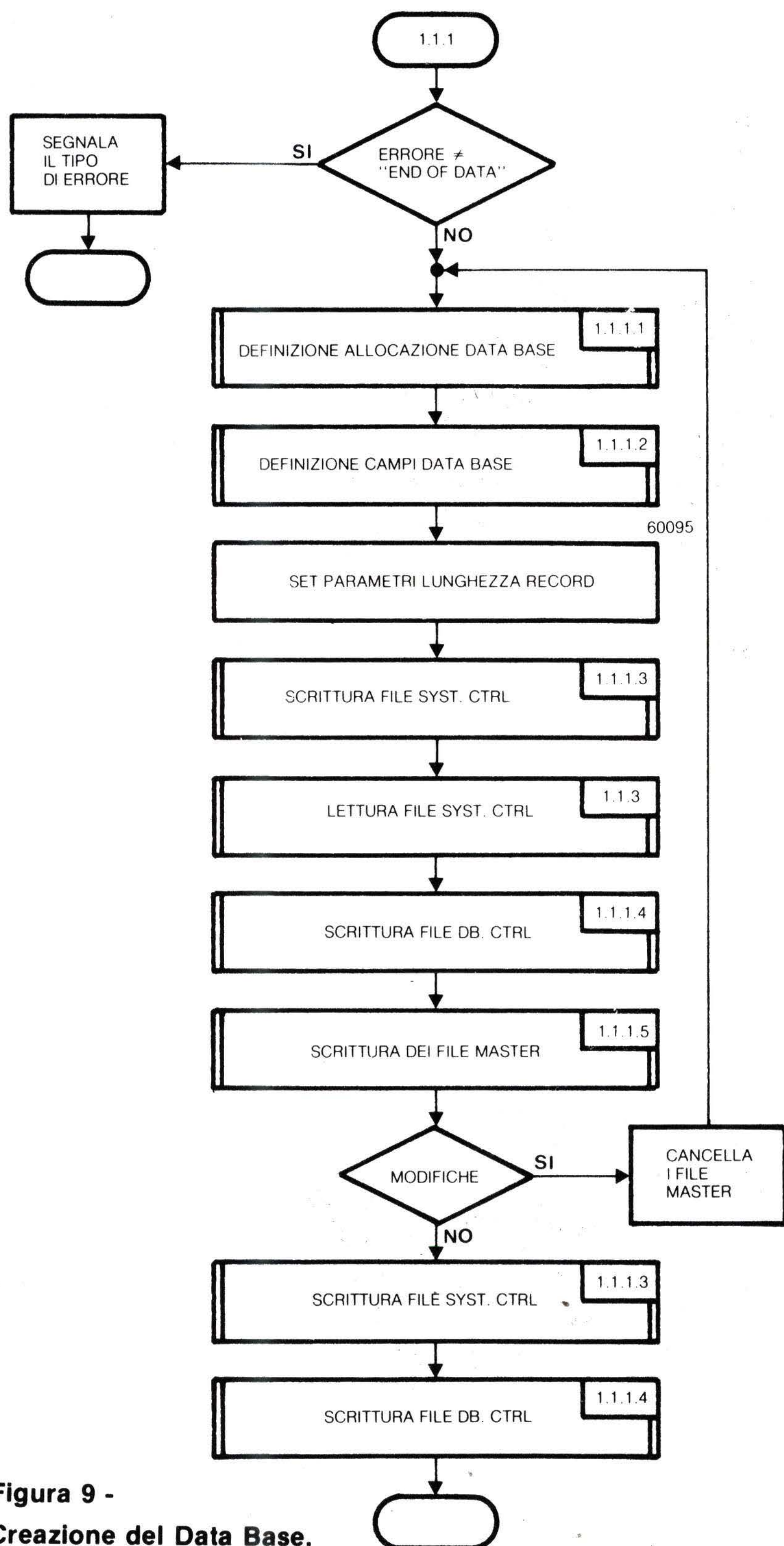
La seconda parte del programma permetterà invece di operare sull'archivio predefinito nelle fasi di:

— inserimento dati
— cancellazione
— ricerca e modifica
— lista
— cambio indice di ricerca

Le due parti costituiranno programmi ben distinti mutuamente richiamabili, poichè condensarli in un unico programma metterebbe in crisi anche la memoria da 48 k dell'Apple.

La *Figura 7* rappresenta come sarà gestito il Data Base. Dopo aver editato i due progressi, mancheranno i files di controllo, per cui al lancio del programma, non esistendo il SYST. CTRL, verrà passato il controllo al programma SETUP che permetterà di configurare il sistema di gestione e successivamente di rilanciare con successo il DATA BASE.
Lo spazio per questo articolo non ci permette di sviluppare sia il SETUP, sia il DB, e, dovendo scegliere, diamo in questa sede spazio al SETUP, in modo da poter trovare nel prossimo articolo le condizioni per caricare il programma di gestione, avendo digerito i flows e listings del SETUP che ci accingiamo a spiegarvi.
Tale modo di procedere è un pò in contraddizione con l'approccio top-down: in effetti, definita la struttura dei dati, trattandosi di due programmi ben definiti, le cui interazioni avvengono solo con la struttura sopra accennata, pensiamo sia più piacevole per il lettore sbilanciarsi a favore di uno sviluppo del SETUP, giungendo fino alla codifica, per ritornare poi in top nel prossimo articolo, completamente dedicato alla gestione della struttura già esistente.
Prima di continuare è necessario rassicurare il lettore sull'hardware necessario: basta un Apple 48K ed almeno un mini floppy montato nello slot 6, drive 1. Ovviamente, se desiderate degli archivi un pò grossi, rischiate di saturare il singolo dischetto, ed in tal caso sarà necessario passare a due o tre unità.

La definizione del Data Base

Il programma di SETUP, seguendo il flow di *figura 8*, inizia con la chiamata ad una routine (62000) di inizializzazione variabili, e con la lettura dei due files di sistema, SYST.CTRL e DB.CTRL. Nelle inizializzazioni compaiono due variabili, RF e RS, che contengono la quantità di memoria libera rispettivamente nel programma DB e nel SETUP: qualora si facciano delle modifiche ai programmi, è necessario aggiornare tali valori.
La prima volta che viene lanciato il programma, questi files non esistono, per cui, invece di proseguire in sequenza ed arrivare al Menu (60500), si attiva la frase ONERR GOTO 60050 che lancia la creazione del Data Base. Tutte le altre volte che verrà eseguito il SETUP, cio non accadrà, se si arriverà direttamente al Menu, che gestisce la possibilità di riconfigurare parzialmente il sistema, o di ampliarlo.
In *Figura 9* possiamo vedere in che modo viene gestita questa creazione iniziale: appena entrati nella routine di errore, ci si chiede se l'errore verificatosi (il cui codice si trova all'indirizzo 222) è quello che ci si aspettava, cioè END OF data, che è il messaggio emesso dal DOS quando si cerca di leggere un

La GenRad Futuredata consegna Intel, Zilog, Motorola, Rockwekk, RCA

8086
8085
8080
z8000
z80
6809
6802
6801
6800
6502
1802
3870

NOI SUPPORTIAMO PIÙ CHIPS
Quando si tratta di incrementare sistemi di sviluppo in grado di supportare più di un microprocessore, nessuno può superarci.
Il nostro sistema universale non vi limita ad una sola famiglia di chip, ma vi permette di progettare con tutti i più noti dispositivi.
Negli anni 80 e nella gara al migliore prodotto, sostituire sistemi di sviluppo sarà senz'altro un inconveniente.
Con il nostro sistema, la sua flessibilità e modularità, questo non sarà necessario.

AGGIUNGIAMO VELOCEMENTE SUPPORTO A NUOVI CHIPS
Grazie al nostro rivoluzionario sistema d'emulazione Slave, siamo in grado di aggiungere supporto a nuovi chips in poche settimane.
Ricordate, noi non fabbrichiamo chips, ma solo sistemi di sviluppo, non dobbiamo riprogettare il sistema per supportare un nuovo componente, ma solo aggiungere una scheda per il suo emulatore; è solo questa che voi pagherete.
Noi offriamo così la soluzione più veloce ed economica.

SEGNAMO IL PASSO DELL'EMULAZIONE
Il nostro sistema è l'unico in grado di offrire una emulazione alla massima velocità senza stati di attesa fino a 10 Mhz.

file inesistente. In caso positivo si prosegue disattivando la frase ONERR (in quanto non più necessaria) con l'istruzione POKE 216,0.

Dopo aver definito dove allocare il Data Base (1.1.1.1) e quali sono i campi richiesti per ogni record Base (1.1.1.2), si possono definire la lunghezza record effettiva R (0) e R (1) dell'INDEX e del Data Base (linea 60095). Possiamo ora scrivere il file SYST.CTRL (1.1.1.3), e rileggerlo (1.1.3) per avere a disposizione le stringhe di comando che la routine di lettura crea; scrivere il DB. CTRL (1.1.1.4) ed infine preparare su disco lo spazio per i files MASTER INDEX (MST.INX) e MASTER FILE (MST.FL) (1.1.1.5).

Dopo questa allocazione di spazio su disco, che può durare qualche minuto, viene presentata la situazione, cioè il numero massimo di records gestibili: se la configurazione non soddisfa, possiamo ancora modificarla (infatti si può ricominciare cancellando i files master e tornando alla definizione della allocazione del Data Base). Se invece tutto va bene, si scrivono i files SYT.CTRL (1.1.1.3) e DB. CTRL (1.1.1.4) definitivi.

In *Figura 10* vediamo come avviene la definizione dell'allocazione del Data Base: dapprima, leggendo i dati (istruzioni DATA che compaiono in fondo al listato) si assumono i parametri (nome file, lunghezza record, slot, drive) di default (linee 62520-62550); in seguito si chiede su quale drive si vuole l'archivio principale del Data Base e si sostituisce questo valore al precedente.

Più complessa è la definizione dei campi all'interno del record (*v. Figura 11*): la routine è strutturata in modo da potere, con le stesse istruzioni, definire un campo, oppure modificare il valore precedente di esso, continuando a rieseguirla finchè non si ottenga una situazione soddisfacente. È importante notare

1.1.1.1 62500

CARICAMENTO PARAMETRI DI DEFAULT

DEFINIZIONE DRIVE DEL DATA BASE

```
62500  REM FIRST TIME
62510  HOME : RESTORE
62520  FOR I = 0 TO 10: READ A$: NEXT
62530  FOR I = 0 TO 10: READ R(I): NEXT
62540  FOR I = 0 TO 10: READ S(I): NEXT
62550  FOR I = 0 TO 10: READ D(I): NEXT
62560  VTAB 9: HTAB 9: INPUT "DATA BASE SU DRIVE: ";A$:A =  VAL (A$)
62570  IF A < > 1 AND A < > 2 THEN 62560
62580  PRINT : HTAB 22: INPUT "SICURO? ";A$
62590  IF  LEFT$ (A$,1) < > "S" THEN 62560
62600  FOR I = 0 TO 2:D(I) = A: NEXT
```

Figura 10 - Definizioni dell'allocazione del Data Base: flow e listato.

che questa definizione, una volta confermata, non sarà più modificabile, a meno di distruggere tutto l'archivio: infatti in questa parte del programma, come visto, si entra se e solo se non esistono i files di sistema, cioè la prima volta che si lancia il programma.

Di ogni campo si definisce:

— Nome: lungo al massimo 9 caratteri,

— Tipo: alfanumerico o numerico,

— Lunghezza: compresa tra 1 e 29.

per i campi di tipo alfanumerico si controllerà poi solo la lunghezza, mentre per quelli di tipo numerico si verificherà anche che siano effettivamente numeri. All'inizio della routine viene chiesto anche il numero di campi che si desidera compongano il record base; il limite massimo di questo valore è stato da noi fissato a 14, ma può essere aumentato a piacere, semplicemente preoccupandosi poi, nel programma DB, di gestire opportunamente l'impaginazione dei dati su video.

Nel calcolo della lunghezza del record, bisogna tenere presente che tra ogni campo all'interno del record esiste il carattere CR separatore, e che quindi la lughezza in bytes è data dalla somma delle lunghezze dei campi più il numero di campi (un CR ognuno). Alla fine della routine si prepara un stringa di caratteri da scrivere sul disco per allocare lo spazio necessario all'archivio. Questa stringa, composta di caratteri non utilizzabili in seguito, è lunga come il record, oppure, se esso è più lungo di 255 bytes, è lunga 255 bytes, essendo questa dimensione sufficiente ad allocare tutto lo spazio necessario.

Nella routine appare una istruzione CALL - 868: è la chiamata ad una funzione di sistema che cancella dal video la riga attuale, dalla posizione del cursore fino alla fine.

Possiamo a questo punto vedere la struttura dei files di sistema SYST.CTRL e DB.CTRL, che contengono i dati di descrizione dell'ambiente in cui operano i due programmi, SETUP e DB.

Nelle *Figure 12 e 13* vediamo la scrittura e la lettura di SYST.CTRL; nelle *Figure 14 e 15* quella di DB.CTRL. In tutto il programma, si vede che la scrittura del file SYST.CTRL è subito seguita dalla routine di lettura, in quanto questa, oltre a leggere i parametri dei vari files, costruisce le stringhe O$ (I), R$ (I), W$ (I), C$ (I) per l'accesso ai files; quindi, ogni volta che si varia uno dei parametri di un file (ad esempio lo slot), bisogna rieseguire questa operazione per potervi accedere.

Proseguendo nell'approfondimento del programma, vediamo nel flow di *Figura 16* come viene allocato lo spazio su disco per i files Master. Inizialmente viene richiesto quanti records si desidera compongano l'archivio base; a questo punto il programma comincia a scrivere sul disco, contemporaneamente sull'INDEX e sul MASTER. Nel primo una chiave fittizia ed il numero d'ordine, nel secondo la stringa preparata nella fase di definizione campi.

Vi sono due possibilità: sul disco scelto vi è spazio sufficiente per contenere l'archivio oppure no. In questa seconda ipotesi, viene attivata la frase ONERR GOTO 60360 contenuta nella linea 60160 e, partendo dall'ultimo record che si è tentato di scrivere e andando all'indietro, si verifica qual'è l'ultima coppia index-master scritta correttamente (ciò perchè il DOS scrive fisicamente sul disco non ad ogni comando di WRITE, ma solo quando ha riempito un buffer di 256 bytes, e quindi a priori non si sa in quale momento si verifica l'errore di disco pieno). In ambedue i casi, arrivando alla linea 60280, in I è contenuto il numero massimo di record Base.

Viene ora calcolato il numero di chiavi contenibili nella memoria del programma DB, perchè in essa poi saranno mantenute per avere un accesso più veloce. Come numero massimo di records utilizzabili dal Data Base nella configurazione scelta, viene quindi scelto il minimo tra il numero di records che sono memorizzabili su disco e il numero di chiavi contenute in me-

1.1.1.2

INPUT N° CAMPI — 63000 63050

STAMPA INTESTAZIONI — 63060

I = 1

INPUT NOME DEL CAMPO I-ESIMO O MODIFICA DEL VALORE PRECEDENTE — 63080 63125

INPUT TIPO DEL CAMPO I-ESIMO O MODIFICA DEL VALORE PRECEDENTE — 63130 63190

INPUT LUNGHEZZA DEL CAMPO I-ESIMO O MODIFICA DEL VALORE PRECEDENTE — 63200 63260

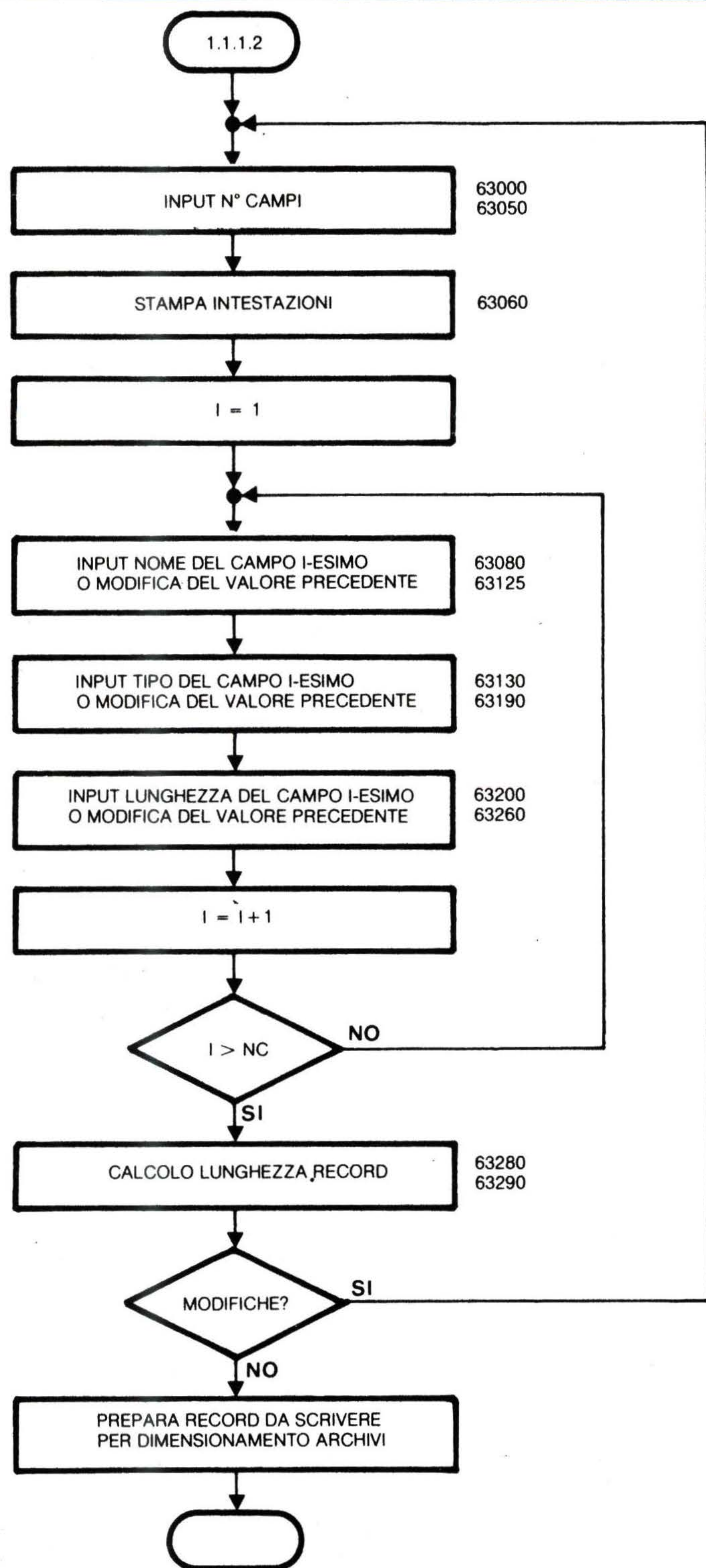
I = I + 1

I > NC — NO / SI

CALCOLO LUNGHEZZA RECORD — 63280 63290

MODIFICHE? — SI / NO

PREPARA RECORD DA SCRIVERE PER DIMENSIONAMENTO ARCHIVI

Figura 11 - Definizione dei campi: flow e listato.

```
63000  REM SETUP
63020  HOME : HTAB 18: PRINT "SETUP"
63030  HTAB 13: PRINT "QUANTI CAMPI? ";: IF NC THEN  PRINT NC;
63035  INPUT "  ";A$: IF A$ <  > "" THEN NC =  INT ( VAL (A$))
63040  IF NC > NM OR NC < 1 THEN 63020
63050  VTAB 2: HTAB 27: CALL  - 868: PRINT NC
63060  VTAB 4: HTAB 11: PRINT "NOME";: HTAB 21: PRINT "TIPO (A/N)";: HTAB
     35: PRINT "LUNGH."
63070  FOR I = 1 TO NC
63080  VTAB I + 5: CALL  - 868: IF I < 10 THEN  HTAB 2
63090  PRINT I;"'CAMPO";
63100  HTAB 10: IF LB$(I) <  > "" THEN  PRINT LB$(I);
63110  HTAB 10: INPUT "";A$
63113  IF A$ = "" AND LB$(I) = "" THEN 63080
63116  IF A$ <  > "" THEN LB$(I) = A$
63120  IF  LEN (LB$(I)) > 9 OR  LEN (LB$(I)) < 1 THEN LB$(I) =  LEFT$ (LB
     $(I),9): GOTO 63080
63125  VTAB I + 5: HTAB 10: PRINT LB$(I)
63130  VTAB I + 5: HTAB 20: CALL  - 868
63140  IF TP(I) = 1 THEN  INPUT "ALFANUMERICO ";A$
63145  IF TP(I) = 2 THEN  INPUT "  NUMERICO ";A$
63150  IF TP(I) = 0 THEN  INPUT "";A$
63160  VTAB I + 5: HTAB 20
63170  IF A$ = "A" THEN TP(I) = 1: CALL  - 868: PRINT "ALFANUMERICO": GOTO
     63200
63180  IF A$ = "N" THEN TP(I) = 2: CALL  - 868: PRINT "  NUMERICO": GOTO
     63200
63190  IF A$ <  > "" OR TP(I) = 0 THEN 63130
63200  VTAB I + 5: HTAB 35: CALL  - 868
63210  IF LL(I) <  > 0 THEN  PRINT  LEFT$ (T$,4 -  LEN ( STR$ (LL(I))));L
     L(I);: HTAB 35
63220  INPUT "";A$
63230  IF A$ = "" AND  NOT LL(I) THEN 63200
63240  IF A$ <  > "" THEN LL(I) =  INT ( VAL ( LEFT$ (A$,3)))
63250  IF LL(I) < 1 OR LL(I) > 29 THEN 63200
63260  VTAB I + 5: HTAB 35: CALL  - 868: PRINT  LEFT$ (T$,4 -  LEN ( STR$
     (LL(I))));LL(I)
63270  NEXT
63280 LR = NC: FOR I = 1 TO NC:LR = LR + LL(I): NEXT
63290  VTAB 21: PRINT "LUNGHEZZA RECORD "LR" BYTES"
63300  VTAB 23: HTAB 20: INPUT "MODIFICHE? ";A$
63310  IF  LEFT$ (A$,1) <  > "N" THEN 63000
63320 ST$ = ""
63330 J = LR - 1: IF J > 255 THEN J = 255
63340  FOR I = 1 TO J:ST$ = ST$ + FR$: NEXT
63350  RETURN
```

1.1.1.3

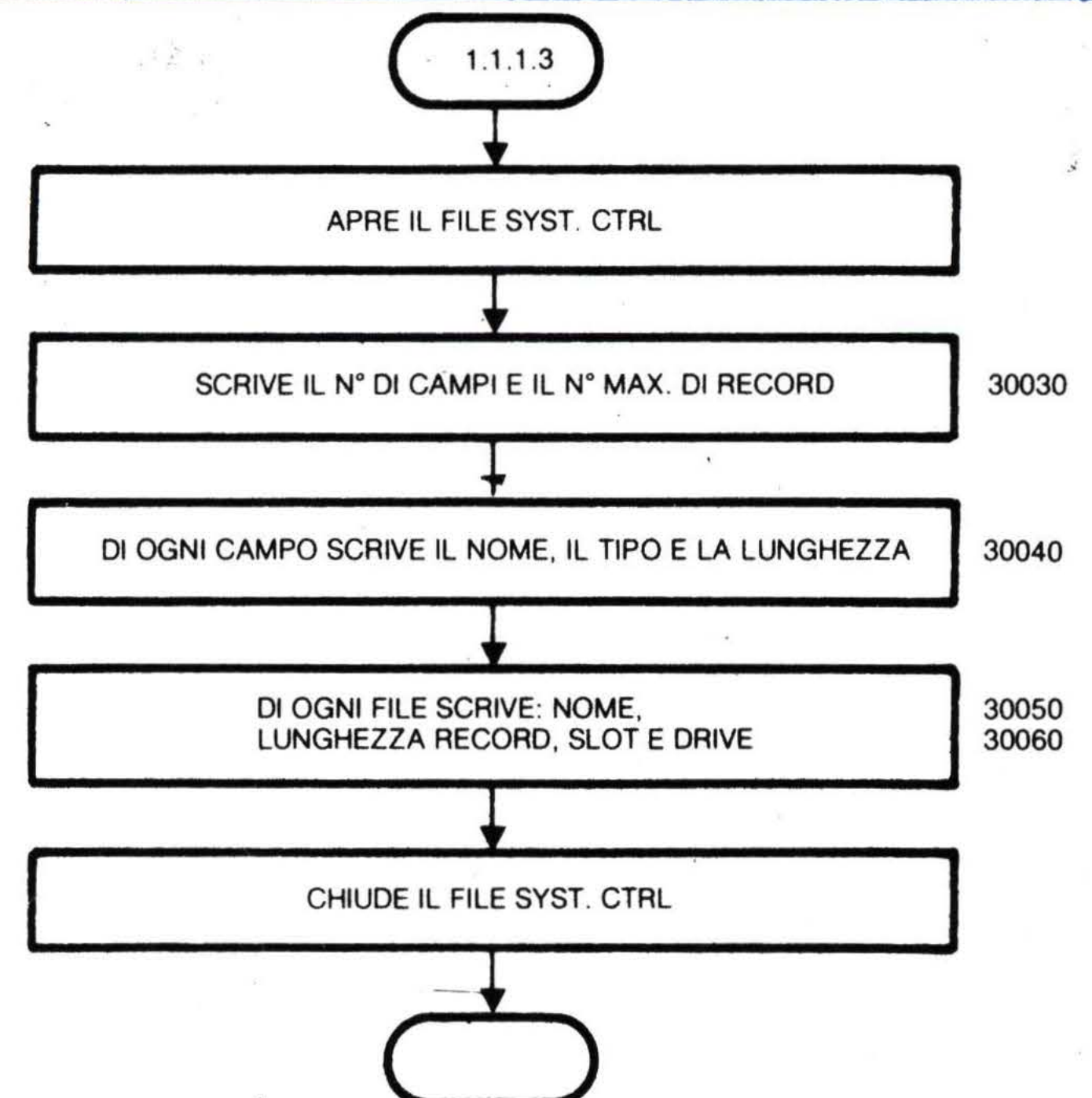
APRE IL FILE SYST. CTRL

SCRIVE IL N° DI CAMPI E IL N° MAX. DI RECORD — 30030

DI OGNI CAMPO SCRIVE IL NOME, IL TIPO E LA LUNGHEZZA — 30040

DI OGNI FILE SCRIVE: NOME, LUNGHEZZA RECORD, SLOT E DRIVE — 30050 30060

CHIUDE IL FILE SYST. CTRL

Figura 12 - Scrittura del file SYST.CTRL: flow e listato.

```
30000  REM WRITE SYSTEM CONTROL
30010  PRINT D$;O1$"SYST.CTRL,S6,D1"
30020  PRINT D$;W1$"SYST.CTRL"
30030  PRINT NC","EM
30040  FOR I = 1 TO NC: PRINT LB$(I)","TP(I)","LL(I): NEXT
30050  RESTORE
30060  FOR I = 0 TO 10: READ A$: PRINT A$","R(I)","S(I)","D(I): NEXT
30070  PRINT D$;C1$"SYST.CTRL"
3008C  RETURN
```

1.1.3

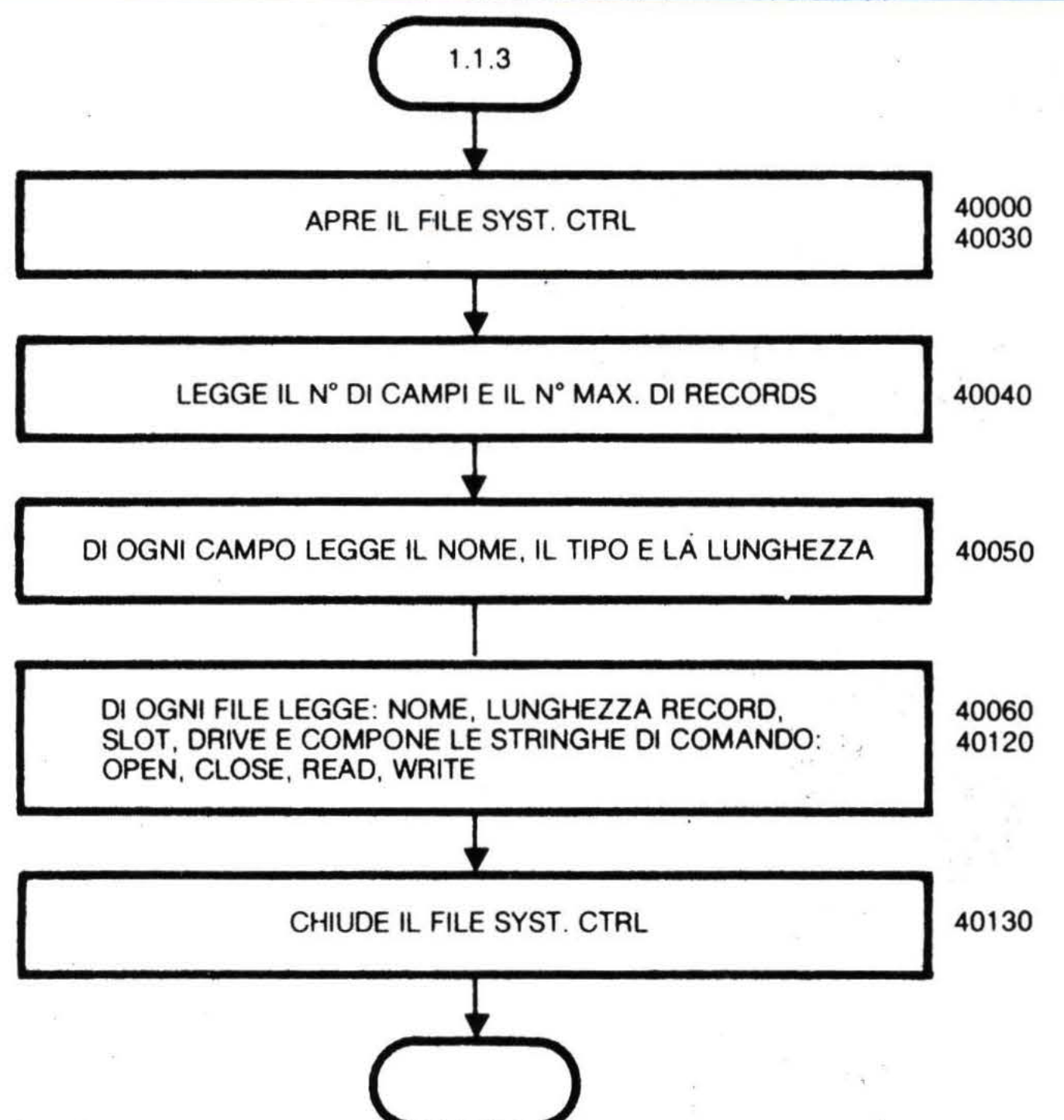
APRE IL FILE SYST. CTRL — 40000 40030

LEGGE IL N° DI CAMPI E IL N° MAX. DI RECORDS — 40040

DI OGNI CAMPO LEGGE IL NOME, IL TIPO E LA LUNGHEZZA — 40050

DI OGNI FILE LEGGE: NOME, LUNGHEZZA RECORD, SLOT, DRIVE E COMPONE LE STRINGHE DI COMANDO: OPEN, CLOSE, READ, WRITE — 40060 40120

CHIUDE IL FILE SYST. CTRL — 40130

Figura 13 - Lettura del file SYST.CTRL: flow e listato.

```
40000  REM READ SYSTEM CONTROL
40010 B$(0) = "":B$(1) = ",R"
40020  PRINT D$;O1$"SYST.CTRL,S6,D1"
40030  PRINT D$;R1$"SYST.CTRL"
40040  INPUT NC,EM
40050  FOR I = 1 TO NC: INPUT LB$(I),TP(I),LL(I): NEXT
40060  FOR I = 0 TO 10
40070 L$ = "": INPUT A$,R(I),S(I),D(I): IF R(I) THEN L$ = ",L" +  STR$ (R
    (I))
40080 O$(I) = O1$ + A$ + L$ + ",S" +  STR$ (S(I)) + ",D" +  STR$ (D(I))
40090 R$(I) = R1$ + A$ + B$(R(I) > 0)
40100 W$(I) = W1$ + A$ + B$(R(I) > 0)
40110 C$(I) = C1$ + A$
40120  NEXT
40130  PRINT D$;C1$"SYST.CTRL"
40140  RETURN
```

ea d'impiego: controlli industriali,

sono disponibili i seguenti moduli hardware novità hardware:	
SMP-E4	**Single board computer espandibile, CPU (8085) + zoccoli per 8/16 KByte EPROM + 2 KByte RAM + USART + 16 linee I/O parallele + 5 Timer/counter (8 bit). Tutte le linee del bus sono bufferizzate rendendo possibile una espansione diretta con tutti i moduli di memoria e di I/O del sistema SMP**
SMP-E8	**CPU (8088) + DMA + 4 KByte RAM + zoccoli per 4/8/16 KByte EPROM. Tutte le linee del bus sono bufferizzate**
SMP-E341	**Floppy Disk controller per minifloppy e standard floppy drives**
SMP-E347	**Magnetic tape controller per mini e standard tape drives**
SMP-E355	**Controller per stampante termica a 20 caratteri**
SMP-E308	**Interfaccia per bus IEC (slave)**
SMP-E309	**Interfaccia per bus IEC (master)**
SMP-E1	Unità centrale (con microprocessore SAB8080) ed ingresso/uscita seriale
SMP-E2	Unità centrale (con microprocessore SAB8085) e controllo del DMA
SMP-E3	Unità centrale (con microprocessore SAB8085) con processore aritmetico e controllo del DMA
SMP-E102-A1	Memoria con 4 KByte RAM statica
SMP-E103-A1	Memoria con 8 KByte RAM statica
SMP-E104	Memoria con 16 KByte RAM statica
SMP-E114	Memoria con 16 KByte RAM dinamica
SMP-E115	Memoria con 32 KByte RAM dinamica
SMP-E120	Memoria con 1 KByte RAM statica e zoccoli per 4 KByte EPROM (8708)
SMP-E121	Memoria con zoccoli per 8 KByte EPROM (8708)
SMP-E122	Memoria con zoccoli per 8 KByte EPROM (2758)
SMP-E123	Memoria con zoccoli per 16/32 KByte EPROM (2716/2732)
SMP-E124	Memoria con 4 KByte RAM statica e zoccoli per 8 KByte EPROM (2758)
SMP-E125	Memoria con 4 KByte RAM statica e zoccoli per 16 KByte EPROM (2716)
SMP-E131	Memoria con 2 KByte RAM CMOS con batteria tampone
SMP-E200	Uscite parallele con livello TTL
SMP-E203	Uscite parallele con 16 relé
SMP-E206	Uscite parallele di potenza con 16 relé (5A-120W)
SMP-E207	Uscite parallele con 24 relé
SMP-E211	Uscite parallele con opto-isolatori
SMP-E212	Ingressi paralleli con opto-isolatori
SMP-E220	Ingresso/uscita seriale con 2 canali
SMP-E230	Ingressi analogici
SMP-E240	Uscite analogiche
SMP-E302	Controllo dell'interrupt (8 livelli) e contatore/timer (6 timer)
SMP-E303	Controllo dell'interrupt (16 livelli) e contatori/timer (10 timer)
SMP-E310	Allacciamento per il pannello di controllo
SMP-S400	Piastra madre
SMP-S410	Adattatore per test
SMP-E420	Alimentatore per ± 5 V e ± 12 V
SMP-E421	Alimentatore per ± 15 V
SMP-E91	Modulo per interfacce speciali realizzate dall'utilizzatore
ES902	Sistema di assemblaggio

Sono disponibili i seguenti moduli software novità software:	
SMP-RTOS	**Sistema operativo in tempo reale**
SMP-STR341	**Driver software per la gestione dei FDD**
SMP-MON1	EPROM (8708) con programma monitor
SMP-BAS1	Set di EPROM con interprete basic

Sono disponibili i seguenti supporti di sviluppo:	
SME FSL 85	**Il sistema di sviluppo SME e la potente biblioteca programmi FSL 85 (routines matematiche, trigonometriche, di regolazione PID ecc.) facilitano lo sviluppo e il test del programma applicativo**

ella Siemens

Data Base personale

```
15700  REM FILE ALLOCATION
15710  VTAB 22: INPUT "OK SLOT, DRIVE? ";A$
15720  IF A$ = "S" OR A$ = "" THEN  RETURN
15730  VTAB 22: CALL  - 958: PRINT "SLOT "S(P1);
15740  HTAB 6: INPUT "";A$:S(P1) =  VAL (A$)
15750  VTAB 22: HTAB 21: PRINT "DRIVE "D(P1);
15760  HTAB 27: INPUT "";A$:D(P1) =  VAL (A$)
15780  HTAB 23: INPUT "MODIFICHE? ";A$
15790  IF A$ = "S" THEN 15730
15800  RETURN
```

Listato della routine di allocazione files.

1.1.1.4

APRE IL FILE DB. CTRL — 42000

SCRIVE: N° RECORDS BASE PRESENTI; N° RECORDS BASE ALLOCATI; FLAG PRESENZA FILE LINK; FLAG TIPO STAMPANTE; FLAG N° COLONNE STAMPANTE; FLAG DISCO PIENO

SCRIVE I FLAG RELATIVI AI 4 POSSIBILI INDICI ACCESSORI

CHIUDE IL FILE DB. CTRL

```
42000  REM WRITE DATA BASE CONTROL
42010  PRINT D$;O$(2)
42020  PRINT D$;W$(2)
42030  PRINT NX","EA","FL","PT","CN","DF
42040  FOR I = 1 TO 4
42050  PRINT IA(I,0)","IA(I,1)","IA(I,2)
42060  NEXT
42070  PRINT D$;C$(2)
42080  RETURN
```

Figura 14 - Scrittura del file DB.CTRL: flow e listato.

1.1.4

APRE IL FILE DB. CTRL — 35000

LEGGE: N° RECORD PRESENTI; N° RECORDS ALLOCATI; FLAG PRESENZA FILE LINK; FLAG TIPO STAMPANTE; N° COLONNE STAMPANTE; FLAG DISCO PIENO

LEGGE I FLAG RELATIVI AI 4 POSSIBILI INDICI ACCESSORI

```
35000  REM READ DB-CONTROL
35010  PRINT D$;O$(2)
35020  PRINT D$;R$(2)
35030  INPUT NX,EA,FL,PT,CN,DF
35040  FOR I = 1 TO 4
35050  INPUT IA(I,0),IA(I,1),IA(I,2)
35060  NEXT
35070  PRINT D$;C$(2)
35080  RETURN
```

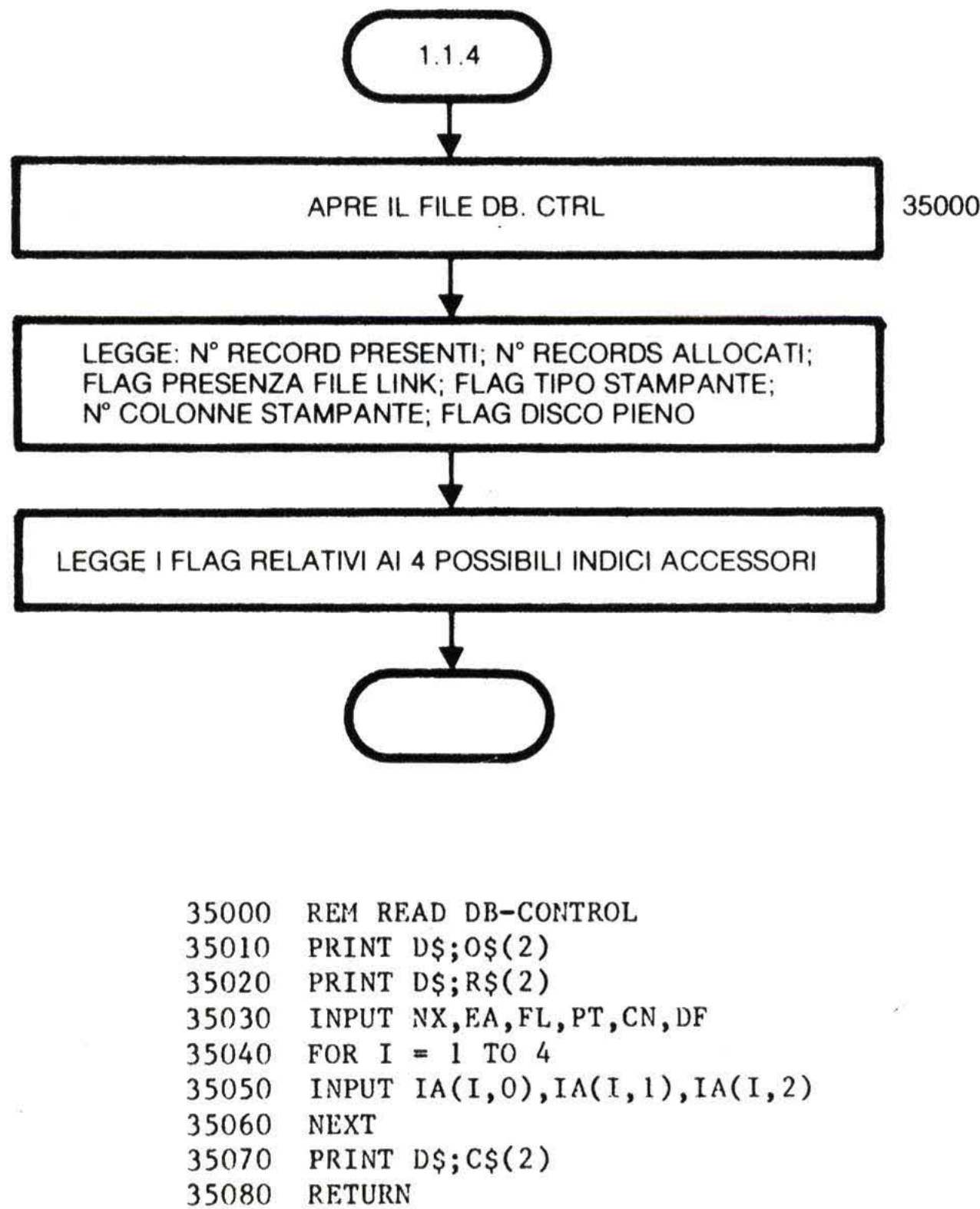



Figura 15 - Lettura del file DB.CTRL: flow e listato.

```
1  REM *********************
2  REM *  A P P L E   I I  *
3  REM *                   *
4  REM * D & V PRODUCTIONS *
5  REM *********************
10  GOTO 60000
500  REM HEAPSORT
510  IF NX = 1 THEN  RETURN
515 L =  INT (NX / 2) + 1:R = NX
520  IF L > 1 THEN L = L - 1:IA% = IX%(L):IA$ = IX$(L): GOTO 540
525 IA% = IX%(R):IA$ = IX$(R):IX%(R) = IX%(1):IX$(R) = IX$(1):R = R - 1
530  IF R = 1 THEN IX%(1) = IA%:IX$(1) = IA$: RETURN
540 J = L
545 I = J:J = 2 * J
550  IF J > R THEN IX%(I) = IA%:IX$(I) = IA$: GOTO 520
555  IF J < R THEN  IF IX$(J) < IX$(J + 1) THEN J = J + 1
560  IF IA$ < IX$(J) THEN IX%(I) = IX%(J):IX$(I) = IX$(J): GOTO 545
565 IX%(I) = IA%:IX$(I) = IA$: GOTO 520
15000  REM INDEX MANAGEMENT
15200  REM DELETE INDEX
15300  REM CREATE INDEX
15700  REM FILE ALLOCATION
20000  REM CREATE NEW LINK FILE
25000  REM PRINTER PARAM.
30000  REM WRITE SYSTEM CONTROL
35000  REM READ DB-CONTROL
40000  REM READ SYSTEM CONTROL
42000  REM WRITE DATA BASE CONTROL
60000  REM MAIN SET-UP
60010  GOSUB 62000
60020  ONERR  GOTO 60050
60030  GOSUB 40000: GOSUB 35000
60040  GOTO 60500
60050  REM START UP
60060  IF  PEEK (222) <  > 5 THEN 63900
60070  POKE 216,0
60090  GOSUB 62500
60095 R(0) = LL(1) + 6:R(1) = LR
60100  GOSUB 30000
60110  GOSUB 40000
60120  GOSUB 42000
60150  REM WRITE MASTERS
60155  VTAB 22: HTAB 9: INPUT "QUANTI RECORDS? ";A$:A =  VAL (A$)
60160  ONERR  GOTO 60360
60190  PRINT D$;O$(0)
60200  PRINT D$;O$(1)
60210  FOR I = 0 TO A
60220  PRINT D$;W$(0);I
60230  PRINT FR$: PRINT I
60240  PRINT D$;W$(1);I
60250  PRINT ST$
60260  NEXT
60270 I = I - 1
60280  PRINT D$;C1$:EM =  INT (RF / (LL(1) + 5)): IF EM > I THEN EM = I: VTAB
  22: PRINT "BUONO!"
60290  VTAB 22: HTAB 9: PRINT "NUMERO MAX DI RECORDS: "EM
60300  VTAB 23: HTAB 31: INPUT "";A$
60310  IF  LEFT$ (A$,1) = "N" THEN  GOSUB 30000: GOSUB 42000: GOTO 60500
60320  IF  LEFT$ (A$,1) <  > "S" THEN 60290
60330  PRINT D$"DELETE MST.INX"
60340  PRINT D$"DELETE MST.FL"
60350  GOTO 60090
60360  IF  PEEK (222) <  > 9 THEN 63900
60370  ONERR  GOTO 60460
60380  PRINT D$;O$(0)
60390  PRINT D$;O$(1)
60400 I = I - 1
60410  PRINT D$;R$(0);I
60420  INPUT A$
60430  PRINT D$;R$(1);I
60440  INPUT A$
60450  POKE 216,0: GOTO 60280
60460  IF  PEEK (222) <  > 5 THEN 63900
60470  PRINT D$;C1$
60480  GOTO 60380
60500  REM  MAIN MENU SETUP
61000  REM RITORNO AL DATA BASE
61010  PRINT D$;C1$
61020  PRINT D$"RUN DB,S6,D1"
62000  REM VARIABLE INIT
62010 D$ =  CHR$ (4)
62020 T$ = "
62030 NM = 14
62040  DIM TP(NM),LL(NM),LB$(NM),LD(1),ID(4,2)
62050 FR$ =  CHR$ (95)
62100 O1$ = "OPEN "
62110 R1$ = "READ "
62120 W1$ = "WRITE "
62130 C1$ = "CLOSE "
62140 RS = 22000
62150 RF = 18000
62160  RETURN
62500  REM FIRST TIME
63000  REM SETUP
63800  DATA "MST.INX","MST.FL","DB.CTRL","LNK.CTRL","LNK.ENTRY","LNK.PTR"
  ,"LNK.FL","INDEX.1","INDEX.2","INDEX.3","INDEX.4"
63810  DATA 1,1,0,0,5,5,1,1,1,1,1
63820  DATA 6,6,6,6,6,6,6,6,6,6,6
63830  DATA 1,1,1,2,2,2,2,2,2,2,2
63900  HOME : FLASH : VTAB 10: HTAB 12: PRINT "CODICE ERRORE = " PEEK (22
  2): NORMAL : END
```

Listato del programma di SETUP.

VARIABILE	DESCRIZ
A	numerico di comodo, usato localmente
A$	var. di comodo, usata molto per input
B$	var. di comodo, usata localmente
B$(I)	vettore di comodo
C$(I)	close file i-esimo
C1$	stringa contenente il valore 'close'
CN	numero max colonne della stampante
D$	variabile stringa con carattere di controllo ctrl-d (CHR$(4))
D(I)	drive di appartenenza del file i-esimo
DF	flag disk full (se 1 indica disco pieno)
EA	indica il numero di record base allocati (comprende anche i cancellati)
FL	flag link (se 1 indica presenza liste)
FR$	variabile contenete CHR$(95) cioe' record libero
I	indice di comodo
IA$	stringa di comodo per algoritmo di sort
IA%	numerico di comodo per algoritmo di sort
IA(I,J)	parametri index i-esimo (J= 0 numero campo, 1 lung. chiave, 2 validita')
IX$(I)	chiave record i-esimo
IX%(I)	indirizzo record i-esimo
J	indice di comodo
L	indice di comodo
LB$(I)	nome campo i-esimo (max 9 caratteri)
LL(I)	lunghezza campo i-esimo (max 29 caratteri)
LR	lunghezza record
NC	numero campi
NM	numero max campi
NX	numero elementi validi dell'archivio base
O$(I)	open file i-esimo
O1$	stringa contenente il valore 'open'
P1	indice di file (parametro per routine allocazione)
PT	tipo interfaccia stampante (0 parallela, 1 seriale)
R	indice di comodo per sort
R$(I)	read file i-esimo
R(I)	parametro indicante lungh. record file i-esimo (0 sequenziale)
R1$	stringa contenete il valore 'read'
RF	memoria libera per chiavi nel programma DB
RS	memoria libera per chiavi nel programma SETUP
S(I)	parametro indicante slot di appartenenza del file i-esimo
ST$	stringa di CHR$(95), utilizzata per allocazione spazi su disco
T$	stringa contenente 20 blank
TP$(I)	tipo campo i-esimo (1 alfanumerico, 2 numerico, 0 non definito)
W$(I)	write file i-esimo
W1$	stringa contenete il valore 'write'

Tabella II - Variabili usate nel programma.

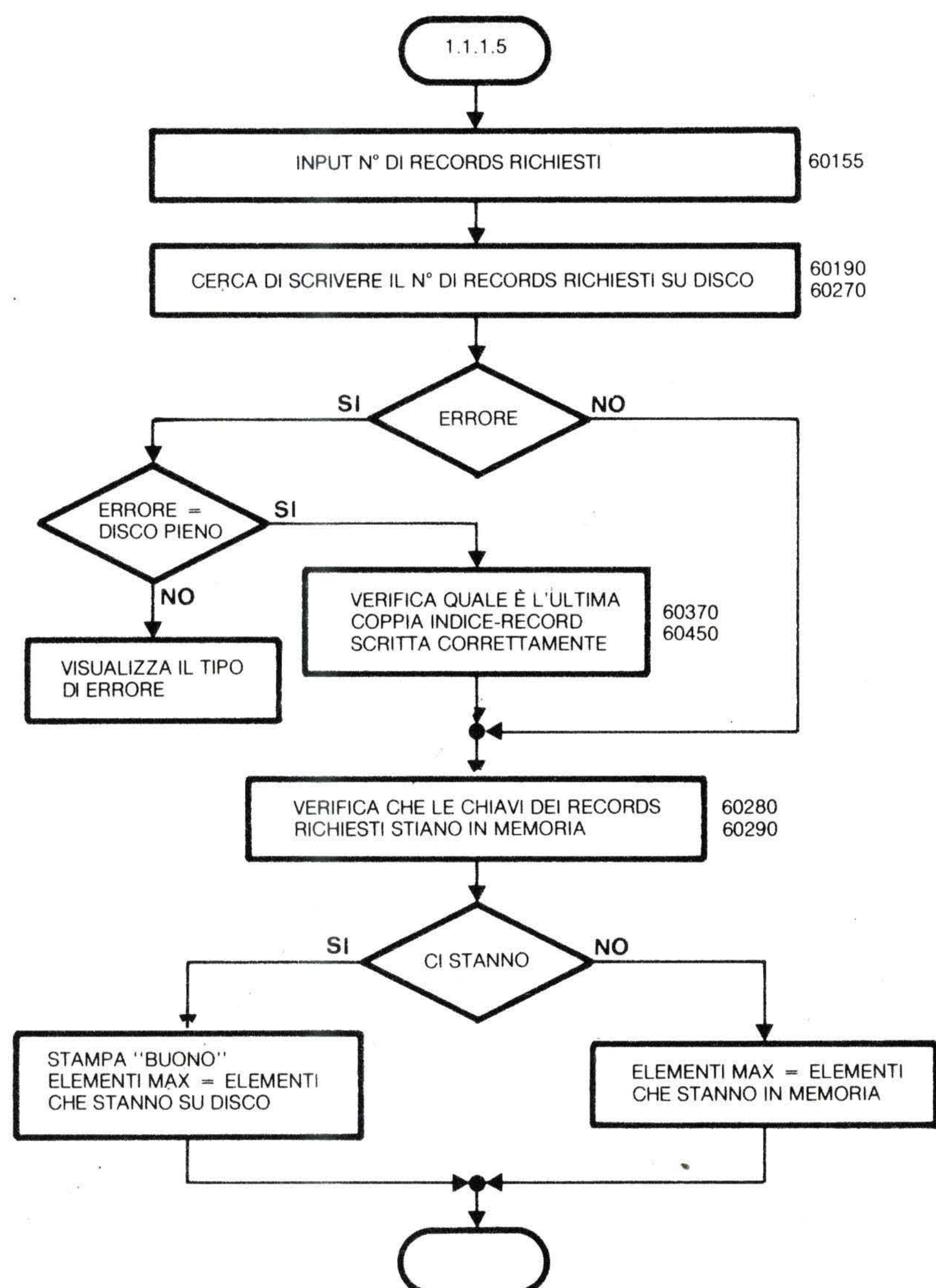
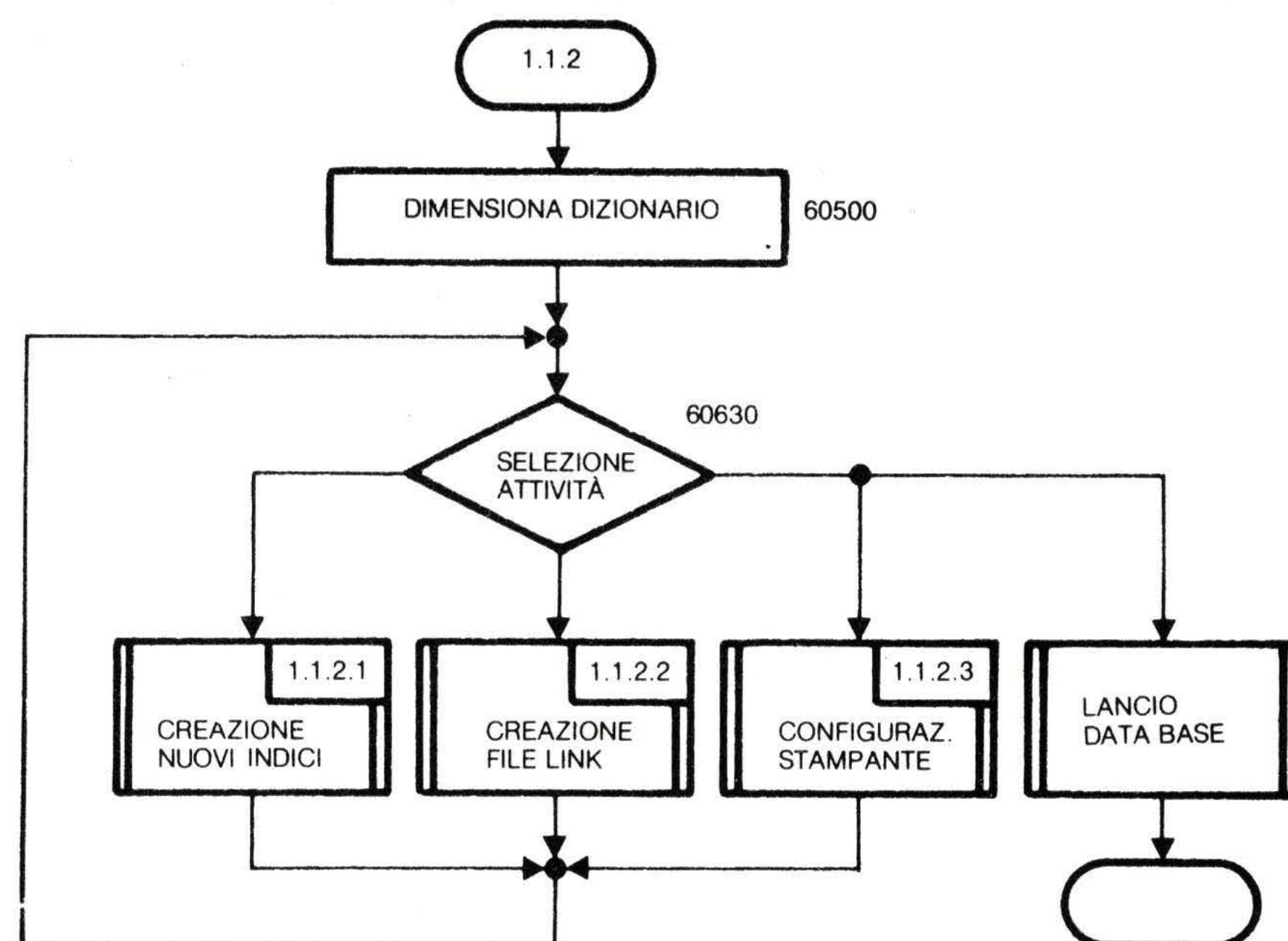
1.1.2
DIMENSIONA DIZIONARIO 60500
60630
SELEZIONE ATTIVITÀ
1.1.2.1 CREAZIONE NUOVI INDICI
1.1.2.2 CREAZIONE FILE LINK
1.1.2.3 CONFIGURAZ. STAMPANTE
LANCIO DATA BASE

```
60500  REM  MAIN MENU SETUP
60505  DIM IX$(EM),IX%(EM)
60510  HOME : HTAB 10: PRINT "RICONFIGURAZIONE SISTEMA"
60520  VTAB 5
60530  PRINT " 1- CREAZIONE NUOVI INDEX"
60540  PRINT " 2- CREAZIONE NUOVO FILE LINK"
60550  PRINT " 3- CONFIGURAZIONE STAMPANTE"
60560  PRINT " 4- RITORNO AL MENU' PRINCIPALE"
60600  VTAB 15: HTAB 10: INPUT "QUALE? ";A$
60610 I =  VAL (A$)
60620  IF I < 1 OR I > 4 THEN 60510
60630  ON I GOSUB 15000,20000,25000,61000
60640  GOTO 60510
```

Figura 17 -Menu del SETUP: flow e listato.

1.1.1.5
INPUT N° DI RECORDS RICHIESTI 60155
CERCA DI SCRIVERE IL N° DI RECORDS RICHIESTI SU DISCO 60190 60270
SI ERRORE NO
ERRORE = DISCO PIENO SI
NO
VISUALIZZA IL TIPO DI ERRORE
VERIFICA QUALE È L'ULTIMA COPPIA INDICE-RECORD SCRITTA CORRETTAMENTE 60370 60450
VERIFICA CHE LE CHIAVI DEI RECORDS RICHIESTI STIANO IN MEMORIA 60280 60290
SI CI STANNO NO
STAMPA "BUONO" ELEMENTI MAX = ELEMENTI CHE STANNO SU DISCO
ELEMENTI MAX = ELEMENTI CHE STANNO IN MEMORIA

Figura 16 - Scrittura del files master.

moria. A questo punto, come visto nel flow di *Figura 9* è ancora possibile fare delle modifiche ricominciando da capo, ad esempio variando la lunghezza della chiave o di alcuni campi, per aumentare il numero di informazioni gestibili.

Il menu principale del SETUP

Abbiamo finalmente terminato di definire la struttura iniziale del Data Base e quindi il programma arriva al Menu principale, come ci arriverà tutte le volte che dal DB chiameremo questo programma per riconfigurare parte del sistema.
Infatti dalla *Figura 17* vediamo che le attività possibili a questo punto sono:

— la gestione di nuovi indici di accesso (1.1.2.1) oltre a quello principale sempre presente ed aggiornato,
— la creazione dei files linkati (1.1.2.2),
— la definizione della stampante collegata (1.1.2.3),
— l'uscita da questo programma col rilancio del DB.

Iniziamo dalla gestione degli indici accessori (*v. Figura 18*): se non sono ancora stati registrati records, non è possibile creare alcun indice, quindi la fase termina immediatamente. Altrimenti viene presentata la situazione attuale, visualizzando, per ognuno dei 4 indici possibili, il nome del campo a cui si riferisce, slot e drive su cui è memorizzato e se è aggiornato oppure no.
Le scelte possibili sono tre: cancellare un indice non più necessario (1.1.2.1.1), crearne uno nuovo (1.1.2.1.2), oppure terminare le operazioni registrando la nuova situazione nei files di sistema.
Per cancellare un indice (*v. Figura 19*) è sufficiente eseguire un comando di DELETE < no-

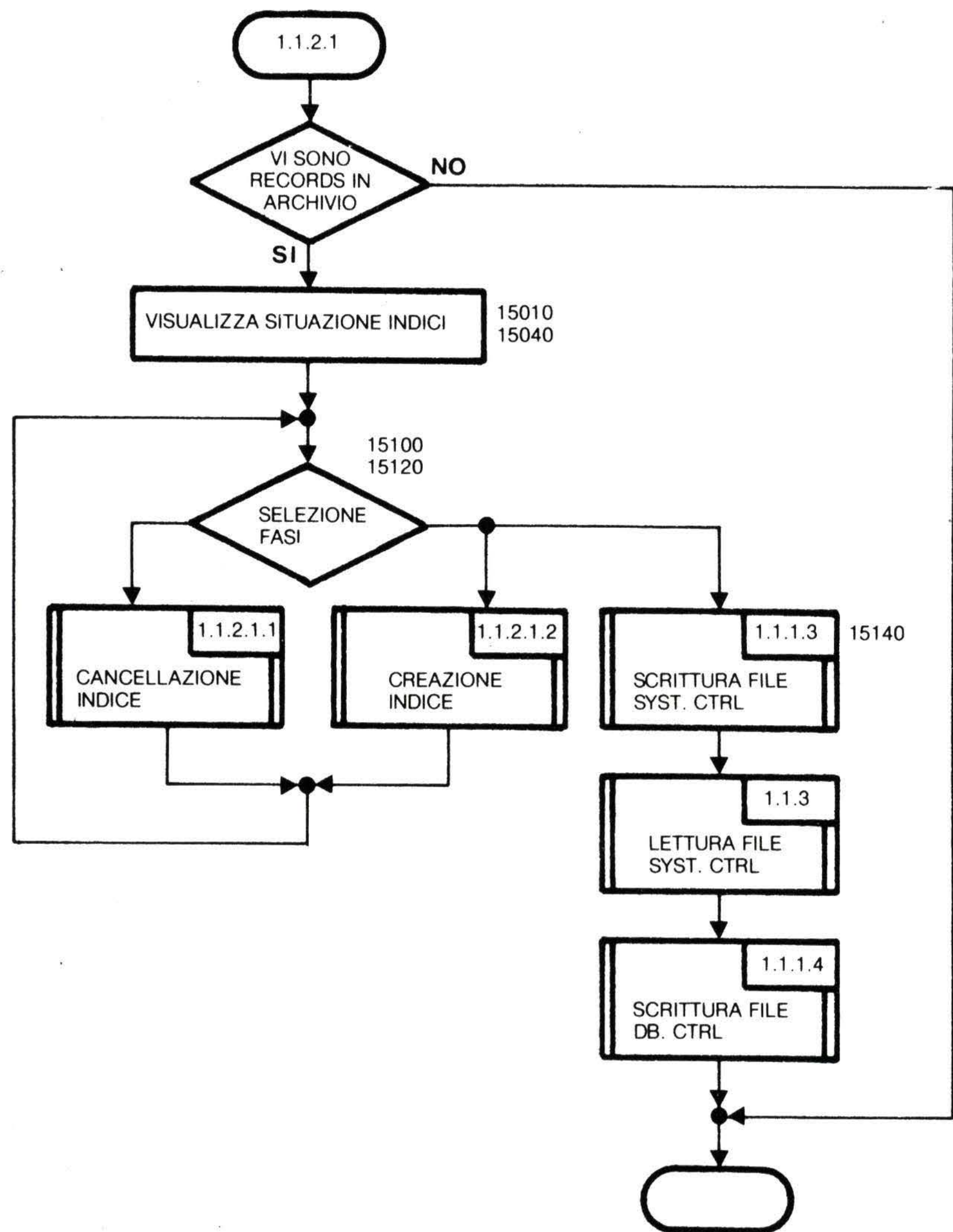
1.1.2.1
VI SONO RECORDS IN ARCHIVIO
NO
SI
VISUALIZZA SITUAZIONE INDICI
15010
15040
15100
15120
SELEZIONE FASI
1.1.2.1.1
CANCELLAZIONE INDICE
1.1.2.1.2
CREAZIONE INDICE
1.1.1.3
SCRITTURA FILE SYST. CTRL
15140
1.1.3
LETTURA FILE SYST. CTRL
1.1.1.4
SCRITTURA FILE DB. CTRL

```
15000  REM INDEX MANAGEMENT
15005  IF  NOT NX THEN  RETURN
15010  HOME : HTAB 15: PRINT "INDICI PRESENTI": PRINT
15020  FOR I = 1 TO 4
15030  PRINT I"  "LB$(IA(I,0));: HTAB 15: PRINT "S"S(I + 6)" D"D(I + 6);:
      IF  NOT IA(I,2) THEN  HTAB 22: PRINT "NON";
15040  PRINT "  AGGIORNATO": NEXT
15060  VTAB 12: PRINT " 1  CANCELLAZIONE"
15070  PRINT " 2  CREAZIONE"
15080  PRINT " 3  FINE"
15100  PRINT : INPUT "QUALE ? ";A$:A% =  VAL (A$)
15110  IF A% < 1 OR A% > 3 THEN 15000
15120  ON A% GOSUB 15200,15300
15130  IF A% <  > 3 THEN 15000
15140  GOSUB 30000: GOSUB 40000: GOSUB 42000
15150  RETURN
```

Figura 18 - Gestione index accesori: flow e listato.

```
15200  REM DELETE INDEX
15210  PRINT : INPUT "QUALE CANCELLARE ? ";A$:A =  VAL (A$)
15220  IF A < 1 OR A > 4 THEN 15200
15230  IF  NOT IA(A,0) THEN  RETURN
15250  PRINT D$"DELETE "; MID$ (O$(A + 6),6,7); RIGHT$ (O$(A + 6),6)
15260 IA(A,0) = 0:IA(A,2) = 0
15270  RETURN
```

1.1.2.1.1
15200
CHIEDE QUALE CANCELLARE
CANCELLA L'INDICE SCELTO
AGGIORNA IL FLAG RELATIVO ALL'INDICE

Figura 19 - Cancellazione di un indice: flow e listato.

me file > ed aggiornare il flag IA (i,j) relativo. Nella linea 15250 si vede che il nome file viene preso dalla stringa O$ relativa all'indice in questione; e dalla stessa stringa vengono presi slot e drive di appartenenza.

Per creare un nuovo indice (*v. Figura 20*) è necessario per prima cosa che non siano già tutti e quattro occupati. Dopo aver chiesto il numero del campo che si vuole utilizzare, viene calcolata la lunghezza massima della chiave contenibile nella memoria del SETUP (linea 15370): siccome le chiavi devono essere contenute in memoria per essere ordinate, bisogna essere sicuri che lo spazio sia sufficiente per tutte. Nella linea 15375 si valorizza la lunghezza del record indice come dimensione della chiave + 6 byte che contengono l'indirizzo del record base relativo ad ogni chiave ed i separatori (CR)

```
15300  REM CREATE INDEX
15310  FOR P = 1 TO 4: IF IA(P,0) THEN  NEXT
15320  IF P > 4 THEN  RETURN
15330  VTAB 18: INPUT "QUALE CAMPO ? ";A$:A =  VAL (A$)
15340  IF A < 2 OR A > NC THEN 15330
15350  PRINT LB$(A);: INPUT "     CONFERMI ?";A$
15355  IF  LEFT$ (A$,1) <  > "S" THEN 15330
15360 IA(P,0) = A
15370 IA(P,1) =  INT (RS / EM) - 5: IF IA(P,1) > LL(A) THEN IA(P,1) = LL(
   A)
15375 R(P + 6) = IA(P,1) + 6:P1 = P + 6: GOSUB 15700: GOSUB 30000: GOSUB
   40000
15380 IA(P,2) = 1
15390  PRINT D$;O$(0)
15400  FOR J = 1 TO NX
15410  PRINT D$;R$(0);J
15420  INPUT A$: INPUT IX%(J)
15430  NEXT
15440  PRINT D$;C$(0)
15450  PRINT D$;O$(1)
15460  FOR J = 1 TO NX
15470  PRINT D$;R$(1);IX%(J)
15480  FOR K = 1 TO A
15490  INPUT A$
15500  NEXT
15505 IX$(J) =  LEFT$ (A$,IA(P,1)): NEXT
15510  GOSUB 500
15520  PRINT D$;O$(P + 6)
15525  PRINT D$;W$(P + 6);0: PRINT NX
15530  FOR J = 1 TO NX
15540  PRINT D$;W$(P + 6);J
15550  PRINT IX$(J): PRINT IX%(J)
15560  NEXT
15570  PRINT D$;C1$
15580  GOSUB 42000
15590  RETURN
```

1.1.2.1.2
15300
15320
VI SONO INDICI DISPONIBILI
NO
SI
CHIEDE IL CAMPO DESIDERATO — 15330 15360
COME LUNGHEZZA MAX. CHIAVE PRENDE LA MINIMA TRA QUELLA DEL CAMPO E LO SPAZIO DISPONIBILE PER OGNI CHIAVE NELLA MEMORIA — 15370
LUNGHEZZA RECORD INDICE = LUNGHEZZA CHIAVE + 6
VERIFICA ALLOCAZIONE FILES
SCRITTURA FILE SYST. CTRL — 1.1.1.3
LETTURA FILE SYST. CTRL — 1.1.3
SETTA IL FLAG DI INDICE PRESENTE
LEGGE GLI INDIRIZZI DEI RECORD BASE VALIDI — 15390 15440
DEI RECORD VALIDI, LEGGE I CAMPI DA PORRE NELL'INDICE — 15450 15505
HEAPSORT — 15510
SCRIVE SUL DISCO IL DIZIONARIO — 15520 15570
SCRITTURA DEL FILE DB. CTRL — 1.1.1.4 — 15580

Figura 20 - Creazione di un indice: flow e listato.

1.1.2.2

DEFINIZIONE SLOT E DRIVE — 20010

SCRITTURA FILE SYST. CTRL — 1.1.1.3 — 20015

LETTURA FILE SYST. CTRL — 1.1.3

PREPARA PARAMETRI PER ROUTINE DI DEFINIZIONE CAMPI — 20020 20050

DEFINIZIONE CAMPI — 1.1.1.2 — 20060

SCRITTURA DEL FILE LINK CTRL — 20070 20140

INIZIALIZZA IL FILE LINK ENTRY — 20150 20210

INIZIALIZZA IL FILE LINK PTR

SETTA IL FLAG DI PRESENZA FILE LINK

SCRITTURA FILE DB. CTRL — 1.1.1.4

LETTURA FILE SYST. CTRL — 1.1.1.3

AGGIUNGE LA LUNGHEZZA RECORD DEL FILE LINK

SCRITTURA FILE SYST. CTRL — 1.1.3

```
20000  REM CREATE NEW LINK FILE
20010  HOME : VTAB 9: PRINT "SLOT "S(3)"  DRIVE "D(3):P1 = 3: GOSUB 15700
     : FOR I = 4 TO 6:S(I) = S(3):D(I) = D(3): NEXT
20015  GOSUB 30000: GOSUB 40000
20020 NC = 0:NM = 8
20030  FOR I = 1 TO NM
20040 LB$(I) = "":TP(I) = 0:LL(I) = 0
20050  NEXT
20060  GOSUB 63000
20070  REM WRITE LINK CONTROL
20080  PRINT D$;O$(3)
20090  PRINT D$;W$(3)
20100  PRINT NC
20110  FOR I = 1 TO NC
20120  PRINT LB$(I)","TP(I)","LL(I)
20130  NEXT
20140  PRINT D$;C$(3)
20150  REM INITIALIZE ENTRY FILE
20160  PRINT D$;O$(4)
20170  FOR I = 0 TO EM
20180  PRINT D$;W$(4);I
20190  PRINT 0
20200  NEXT
20210  PRINT D$;C$(4)
20220  PRINT D$;O$(5)
20230  PRINT D$;W$(5);0
20240  PRINT 1
20250  PRINT D$;C$(5)
20260 FL = 1:DF = 0
20270  GOSUB 42000
20280  GOSUB 40000
20285 R(6) = LR: GOSUB 30000
20290  RETURN
```

Figura 21 - Creazione del file LINK: flow e listato.

```
25000  REM PRINTER PARAM.
25010  HOME : HTAB 12: PRINT "STAMPANTE TIPO:"
25020  VTAB 5: HTAB 5: IF  NOT PT THEN  PRINT "PARALLELA";: GOTO 25030
25025  PRINT "SERIALE";
25030  HTAB 20: PRINT "DA "CN" COLONNE"
25040  VTAB 15: HTAB 20: INPUT "MODIFICHE? ";A$: IF  LEFT$ (A$,1) <  > "S
     " THEN 25100
25050  VTAB 18: INPUT "STAMPANTE (S/P)? ";A$:PT = 0
25060  IF A$ = "S" THEN PT = 1
25070  INPUT "NUMERO COLONNE? ";A$:CN =  INT ( VAL (A$))
25080  GOTO 25000
25100  GOSUB 42000
25110  RETURN
```

1.1.2.3

VISUALIZZA SITUAZIONE ATTUALE — 25000 25050

MODIFICA DEI PARAMETRI

SCRITTURA FILE DB. CTRL — 1.1.1.4

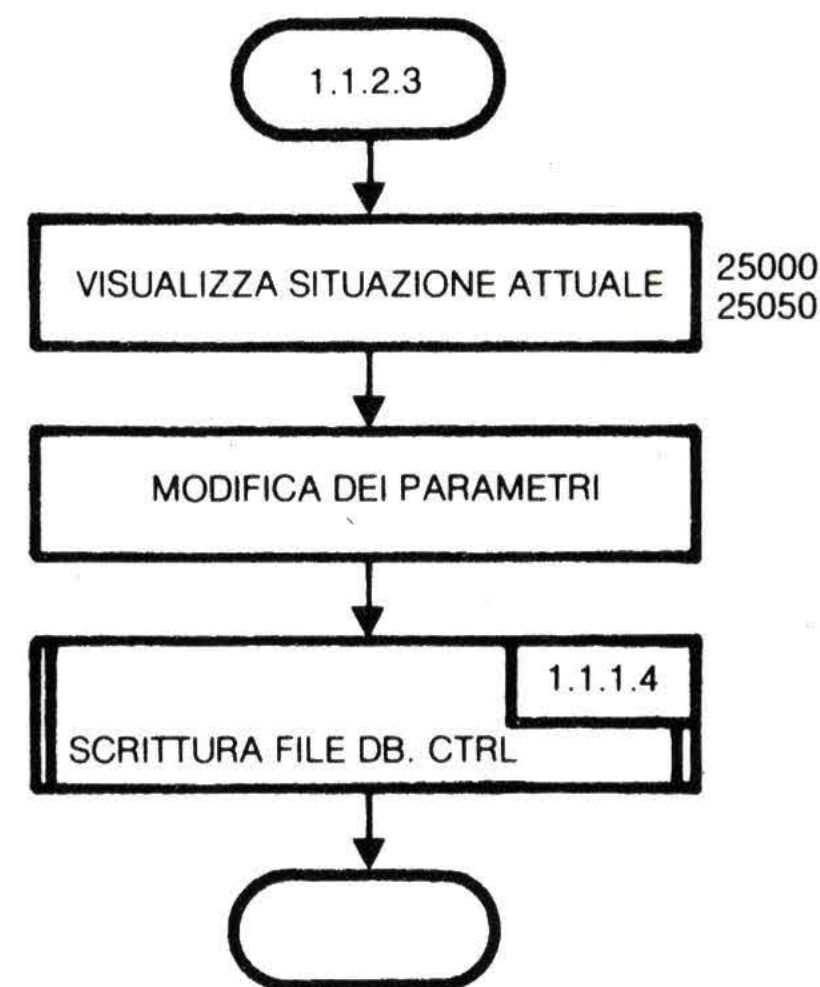



Figura 22 - Configurazione della stampante: flow e listato.

tra chiave ed indirizzo.
Subito dopo viene eseguita una routine (che inizia alla linea 15700) che permette di variare lo slot ed il driver su cui memorizzare l'indice. È a cura dell'operatore non mettere gli indici su di un disco che non abbia lo spazio libero necessario per contenerli: altrimenti, durante la scrittura dell'indice si verificherà l'errore di 'DISK FULL' (in ogni caso potrà sempre ricrearlo su di un altro disco). Scrivendo e rileggendo il file SYST.CTRL, si hanno a disposizione le stringhe di comando aggiornate per l'accesso ai dati.
Per creare l'indice richiesto, a questo punto è necessario leggere dal MASTER INDEX quali sono i records dell'archivio validi, memorizzando questi indirizzi nel vettore IX% (linee 15390-15440: viene letta anche la chiave principale nella variabile A$, ma non essendo necessaria, non viene memorizzata). Sapendo ora quali sono i records validi del MASTER FILE, possiamo leggerli uno per volta e memorizzare nel vettore IX$ il campo relativo all'indice in creazione e scartando gli altri campi. Alla fine di questa scansione abbiamo nei vettori IX$ e IX% tutte le informazioni necessarie per accedere ai records secondo un altro indice; non resta che ordinarli con la routine di *heapsort* già vista nel numero scorso, per poter effettuare la ricerca binaria; memorizzarli sul disco scelto e registrare nel DB.CTRL la presenza di un nuovo indice aggiornato.
Il flow di *Figura 21* rappresenta la fase di creazione dei files LINK, che inizia con la definizione di quale slot e drive si vuole usare per questi dati e quindi crea le stringhe di comando col solito procedimento di scrittura e rilettura del file SYST.CTRL.
Per definire i campi che si desidera compongano il record linkato, si utilizza la stessa procedura usata per il record base, semplicemente variando alcuni parametri, ed utilizzando le stesse variabili (per cui dopo ciò sarà necessario rileggere il file SYST.CTRL per ripristinare la situazione precedente). La definizione dei campi crea parametri simili a quelli del record principale, che vengono scritti nel file LNK. CTRL. L'inizializzazione degli altri files necessari alla gestione dei link consiste semplicemente nell'azzerare tutto il file LNK.ENTRY, a significare che non vi sono liste presenti (infatti le abbiamo solo definite), e nello scrivere nel file LNK.PTR qual è il primo record libero, cioè il numero 1. La conclusione della fase è come al solito la scrittura dei files di sistema per aggiornare la situazione.
L'ultima fase (*v. Figura 22*) riguarda la definizione del tipo di stampante che si vuol usare. La differenza è nel tipo di interfaccia (che deve essere nello slot n. 1). Vi sono due possibilità: parallela o seriale. L'altro parametro è il numero massimo di caratteri che la stampante gestisce per ogni riga (tipicamente 80 o 132). Anche questi parametri vengono registrati nel file DB.CTRL.

Conclusione

Abbiamo visto come creare la strutura del Data Base, come creare i files linkati e gli indici accessori, quali sono i parametri necessari alla gestione di tutto l'insieme, e come configurare il sistema secondo le proprie esigenze e le possibilità della propria macchina (ad esempio il numero di drives che si possiedono).
Nel prossimo numero proseguiremo con lo sviluppo, analizzando il programma vero e proprio di gestione dati. ■

CORSI SPERIMENTALI S

5 giorni con il microcomuter in collab
SERVIZIO DI TRADU

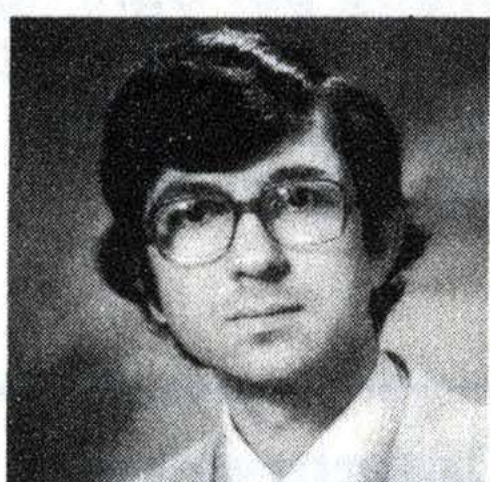
LA MICROLEM, PRIMA AZIENDA ITALIANA AD ORGANIZZARE CORSI «HANDS -ON» SUI MICROCOMPUTER E L'ELETTRONICA DIGITALE, A CONFERMA DEL SUCCESSO DI QUESTA FORMULA DI ADDESTRAMENTO PRATICO ED ALTAMENTE PROFESSIONALE,

ORGANIZZATI IN COLLABORAZIONE CON «LA SCUOLA DI ELETTRONICA». ISTRUTTORI QUALIFICATI ED ATTREZZATURE DIDATTICHE ADEGUATE GARANTISCONO AI PARTECIPANTI UN RAPIDO APPRENDIMENTO DEI PRINCIPI FONDAMENTALI DEI MICROCOMPUTER E DELLE LORO POTENZIALI APPLICAZIONI

10° CORSO 22 - 27 SETTEMBRE	DALL'ELETTRONICA DIGITALE AL MICROPROCESSORE: PROGRAMMAZIONE E INTERFACCIAMENTO CORSO RISERVATO AGLI INSEGNANTI DEGLI ISTITUTI TECNICI

Dr. PETER R. RONY. *Il dr. Peter R. Rony è professore presso il dipartimento di ingegneria chimica del Virginia Polytechnic Institute & State University di Blacksburg, Virginia. E' l'ideatore e l'autore principale dei BUGBOOKS sull'elettronica digitale e sui microprocessori, veri e propri libri di testo, tradotti ormai in molte lingue fra cui l'italiano. Il prof. Rony si può definire un «docente per eccellenza» grazie alla sua particolare chiarezza di esposizione e alle sue tecniche di insegnamento assolutamente innovative. Egli è molto noto in Italia per aver condotto i primi corsi sperimentali sui microprocessori e sull'elettronica digitale.*

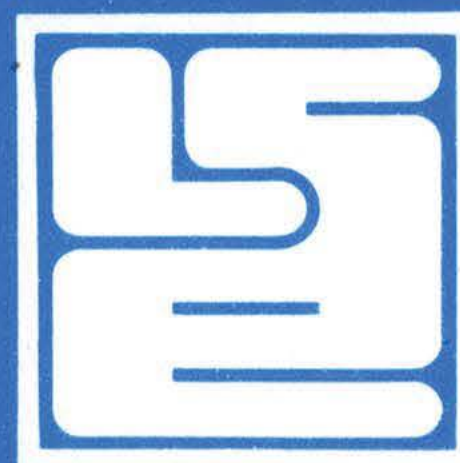
DESCRIZIONE DEL CORSO: due giorni sono dedicati ad un ripasso, con l'esecuzione di molti esperimenti, dell'elettronica digitale, dalla semplice porta logica alle memorie a semiconduttore. I quattro giorni sucessivi sono dedicati allo studio del microprocessore di cui si esaminano, verificandole sperimentalmente, le caratteristiche di programmazione e di interfacciamento dopo aver definito i concetti fondamentali, da quello di *bit* a quello di *bus* a quello di *programma*, di *dispositivo I/O*, di *ciclo macchina*, *set di istruzioni*, *flag*, *interrupt*, ecc. Viene fatto un uso molto esteso di diapositive, in lingua italiana, che potranno essere eventualmente richieste dagli insegnanti per l'uso durante le loro lezioni scolastiche. **CHE COSA SI IMPARA:** ciò che serve ad ogni buon insegnante: trasferire all'allievo i concetti fondamentali dell'elettronica digitale e della programmazione e interfacciamento dei microcomputer, verificandoli sperimentalmente su attrezzature didattiche appositamente realizzate, e stimolandone il più vivo interesse. **CHI DOVREBBE SEGUIRE IL CORSO:** tutti gli insegnanti che desiderano perfezionare le proprie tecniche di insegnamento, dando vita ad un rapporto più vivo e nuovo con l'allievo. Come libri di testo vengono adottati i BUGBOOK V & VI in edizione italiana. Ad ogni coppia di partecipanti viene dato in uso un microcomputer didattico MMD1 con gli accessori e componenti per l'esecuzione degli esperimenti di programmazione e di interfacciamento. **CHIUSURA DELLE ISCRIZIONI:** 13 settembre 1980. **MASSIMO NUMERO DI PARTECIPANTI AMMESSO:** 36. L'entità della quota di iscrizione verrà comunicata su specifica richiesta dei Presisidi di Istituto, secondo il numero di insegnanti che l'Istituto intende iscrivere.

11° CORSO 13 - 17 OTTOBRE	MICROPROCESSORI: CONCETTI FONDAMENTALI E APPLICAZIONI CORSO SPERIMENTALE DEL MASSACHUSSETS INSTITUTE OF TECHNOLOGY

CHRISTOPHER E. STRANGIO. *Ha conseguito i titoli di S.M. in ingegneria elettronica ed informatica e di E.E. presso il Massachussets Institute of Technology nel 1976. Ha eseguito un lavoro di ricerca presso il M.I.T. relativo alla verifica automatica dell'autenticità delle firme per mezzo di un computer. Come responsabile per lo sviluppo dei corsi presso la Lowell Institute School del M.I.T., ha maturato una notevole esperienza di insegnamento. Per conto del M.I.T. ha realizzato su videocassette un corso di 20 ore sui microprocessori. E' autore del libro «Digital electronics: fundamental concepts and applications», pubblicato da Prentice-Hall.*

DESCRIZIONE DEL CORSO: è suddiviso in 10 sezioni, ciascuna delle quali è costituita di una lezione di circa un'ora e mezzo e di due ore di laboratorio. Vengono esaminati sia l'aspetto *hardware* che *software* dei sistemi a microprocessore, con particolare riguardo all'8080, che è ancora il più usato. Circa metà del tempo è dedicato ad *esperimenti di programmazione* da eseguire sui microcomputer messi a disposizione in ragione di uno ogni coppia di partecipanti. Una «Study Guide» e un «Laboratory Manual» riproducono tutte le diapositive utilizzate nelle lezioni, una sintesi di queste ultime, oltre a problemi supplementari, tests per autoistruzione e le soluzioni di tutti gli esercizi di laboratorio. Ogni sera viene assegnato ai partecipanti un lavoro di lettura e preparazione per circa due ore. E' un corso intensivo, che richiede il massimo impegno. **CHE COSA SI IMPARA:** i concetti fondamentali della microprogrammazione, della logica programmata e delle sue applicazioni pratiche. **CHI DOVREBBE SEGUIRE IL CORSO:** i dirigenti d'azienda, piccoli imprenditori, progettisti di controlli industriali e tutti coloro che desiderano conoscere quali *effettivi vantaggi* derivano in termini pratici dall'estesa utilizzazione dei microprocessori. Ad ogni partecipante vengono forniti libri di testo ed altra documentazione in lingua inglese. Ad ogni coppia di partecipanti viene dato in uso un microcomputer didattico MMD1 appositamente attrezzato per conto del M.I.T. **CHIUSURA DELLE ISCRIZIONI:** 20 settembre 1980. **MASSIMO NUMERO DI PARTECIPANTI AMMESSO:** 24. **COSTO:** L. 550.000 + IVA.

PER ISCRIZIONI E INFORMAZIONI PIU' DETTAGLIATE RIVOLGETEVI A : «LA SCUOLA

SUI MICROPROCESSORI

orazione con «La Scuola di Elettronica»
JZIONE SIMULTANEA

12° CORSO 27 - 31 OTTOBRE	**PROGRAMMAZIONE E INTERFACCIAMENTO DEI MICROPROCESSORI** CORSO DI INTRODUZIONE AL MICROCOMPUTER ORGANIZZATO PER CONTO DEL VIRGINIA POLYTECHNIC INSTITUTE & STATE UNIVERSITY

DAVID G. LARSEN. *E' istruttore di Elettronica presso il dipartimento di chimica del Virginia Polytechnic Institute & State University di Blacksburg, Virginia, dove svolge attività didattica a vari livelli nel campo dell'elettronica analogica e digitale. E' coautore dei BUGBOOKS ed è stato l'inventore degli utilissimi «outboards» per la rapida esecuzione degli esperimenti di elettronica digitale. Mr. Larsen tiene regolarmente corsi brevi sui microprocessori per conto della Extension Division del V.P.I. Per questa sua attività è assai noto anche in Italia. Collabora alla redazione dei «Columns», articoli sui microprocessori, pubblicati da note riviste, anche italiane.*

DESCRIZIONE DEL CORSO: circa un terzo del corso è dedicato al ripasso dei concetti fondamentali di elettronica digitale ed alla loro verifica sperimentale. I rimanenti due terzi sono dedicati ai problemi della microprogrammazione e a quelli relativi all'interfacciamento del microprocessore col mondo esterno. L'esposizione dei concetti fondamentali della programmazione viene immediatamente integrata con la sperimentazione. I problemi dell'interfacciamento, che sono quelli di più immediato interesse pratico, vengono affrontati in modo graduale e risolti in modo chiaro: ne viene quindi fatta la verifica sperimentale. **CHE COSA SI IMPARA:** il corso garantisce l'apprendimento dei fondamenti dell'elettronica programmata e fornisce le basi necessarie per poter proseguire nella materia in modo autonomo. **CHI DOVREBBE SEGUIRE IL CORSO:** coloro che vogliono rendersi conto di quale *effettivo e reale* potenziale sta dietro il microprocessore dal punto di vista applicativo e coloro che, assolutamente digiuni in materia, hanno l'esigenza di affrontare il microprocessore dal punto di vista progettuale pratico. Ad ogni partecipante viene data una copia personale dei testi in lingua italiana adottati durante il corso. Ad ogni coppia di partecipanti viene dato in uso un microcomputer didattico MMD1con tutti i componenti necessari per l'esecuzione degli esperimenti. **CHIUSURA DELLE ISCRIZIONI:** 3 ottobre 1980. **MASSIMO NUMERO DI PARTECIPANTI AMMESSO:** 32. **COSTO:** L. 500.000 + IVA.

13° CORSO 17 - 21 NOVEMBRE	**COME CONOSCERE E USARE IL MICROPROCESSORE 8085** CORSO SPERIMENTALE REALIZZATO SUL MICROCOMPUTER SDM85

Dr. LANCE A. LEVENTHAL. *Dopo aver conseguito il titolo di Ph.D. in fisica applicata e scienza dell'informazione nel 1973 presso l'Università di California, San Diego, ha lavorato presso università ed aziende statunitensi, sviluppando una notevole esperienza nella progettazione di apparecchiature a microprocessore. E' presidente della Emulative Systems Co., da lui fondata nel 1975, società di consulenza specializzata nei settori applicativi dei microprocessori e microcomputer. E' molto noto come autore di alcuni ottimi libri sui microprocessori e di oltre 40 articoli, comparsi sulle maggiori riviste americane.*

DESCRIZIONE DEL CORSO: il corso esamina in dettaglio l'architettura e il set di istruzioni del microprocessore 8085. Di esso vengono descritti gli aspetti *hardware*, in particolare quelli relativi ai *segnali di controllo, I/0, interrupts e memoria,* dopo aver fornito definizioni e concetti fondamentali. Ogni argomento svolto viene verificato sperimentalmente. Particolare attenzione è riservata ad esperimenti sugli I/0 e sul timing, all'uso pratico dei chips di supporto, ed a specifiche applicazioni sperimentali. Si introduce il concetto di *sistema di sviluppo,* si discute la formulazione di un problema, l'integrazione «hardware-software» e il testing. Si esaminano quindi applicazioni pratiche e si fanno considerazioni sugli sviluppi futuri dei microprocessori. **CHE COSA SI IMPARA:** il corso garantisce un soddisfacente apprendimento delle caratteristiche dettagliate di un particolare tipo di microprocessore e di tutte quelle nozioni di base necessarie ad un potenziale utilizzatore. Il corso fornisce informazioni utili che possono essere utilizzate anche da coloro che abitualmente usano altri tipi di microprocessori. **CHI DOVREBBE SEGUIRE IL CORSO:** coloro che hanno responsabilità decisionali circa l'utilizzazione dei microprocessori nelle applicazioni industriali e tutti i tecnici e progettisti che intendono approfondire le proprie conoscenze nel campo della progettazione a microprocessore. Per seguire il corso col massimo profitto è raccomandabile una buona conoscenza dei concetti fondamentali di elettronica digitale. Ad ogni partecipante viene rilasciata una copia personale del libro «Microcomputer Experimentation with the INTEL SDK 85» scritto dallo stesso docente e che servirà da guida durante il corso. Come sussidio didattico verrà utilizzato il microcomputer SDM 85. **CHIUSURA DELLE ISCRIZIONI:** 25 ottobre 1980. **MASSIMO NUMERO DI PARTECIPANTI AMMESSO:** 30. **COSTO:** L. 600.000 + IVA.

14° CORSO 9 - 11 DICEMBRE	**CHIPS DI INTERFACCIA PROGRAMMABILI E INTERFACCIAMENTO ASINCRONO SERIALE** CORSO AVANZATO, PER AMPLIARE LE PROPRIE CONOSCENZE SUI MICROPROCESSORI

Dr. PETER R. RONY. «Advanced microcomputer interfacing with Programmable Peripheral Interface chips and asynchronous serial interfacing» è il titolo esatto di questo corso alla cui preparazione il Prof. Rony ha dedicato molto tempo e particolare attenzione. Il corso risponde in sostanza al quesito «Quali reali possibilità esistono oggi per la realizzazione pratica di circuiti di interfaccia fra il microprocessore e reali circuiti di controllo industriale?». Esiste oggi un'intera famiglia di «chips» di interfaccia, detti brevemente PPI - Programmable Peripheral Interface -, che consentono brillanti soluzioni di interfaccia, molto vantaggiose per affidabilità, elasticità di applicazione e costo. L'analisi di questi «chips», fra cui l'8251, l'8253, l'8255, l'8259' sarà l'oggetto di questo corso avanzato. Come al solito, verranno eseguiti molti esperimenti al fine di evidenziare le reali possibilità applicative di questi chips. Ad ogni partecipante verrà consegnata una copia personale del BUGBOOK IV in edizione originale americana; verrà inoltre dato il libro «An introduction to microcomputers Vol. II» della Osborne ed ampia letteratura tecnica ed applicativa. Questo è un corso avanzato per cui si raccomanda la partecipazione a tutti quei tecnici e progettisti in possesso di almeno qualche conoscenza relativa al set di istruzioni ed alle possibilità di interfacciamento dell'8080.
L'abilità e la chiarezza del Prof. Rony garantiscono un soddisfacente apprendimento dell'argomento trattato, tanto importante oggi dal punto di vista applicativo. **NUMERO MASSIMO DI PARTECIPANTI:** 24. **COSTO:** (indicativo) L. 480.000 + IVA.

DI ELETTRONICA» VIA VITTOR PISANI, 22 - 20124 MILANO - TEL. (02) 6572815 - 6573050

di **F. Waldner** - *Università di Bari, Istituto di Fisica*

Parte IV

In questo articolo saranno trattati i *tipi speciali* introdotti e definiti dal programmatore: in Pascal possono essere di vari generi, in modo da dare all'utente uno strumento di incredibile flessibilità per l'organizzazione ed il trattamento dei dati.

Tipi scalari speciali

Riprendiamo il discorso lasciato in sospeso nella Parte II, ricordando che ci eravamo limitati ad esaminare i tipi standard (*reale, intero, booleano, carattere*) mentre vi avevo solo annunciato l'esistenza di altri tipi, la cui definizione veniva lasciata all'utente. La cosa vi sarà magari sembrata un pò misteriosa: è quindi giunto il momento di spiegare di che cosa si tratta con un esempio.
Supponete di voler classificare mese per mese della corrispondenza o delle fatture messe alla rinfusa. Ciò che normalmente si fa è di perforare delle opportune schede, il cui formato viene studiato sulla base del problema, di farle leggere al computer, e di far girare un programma che selezioni tutte le schede di gennaio, di febbraio, etc. Se riserviamo tre colonne al mese, in queste colonne viene perforato qualcosa come:

```
GEN, FEB, MAR, APR, MAG, GIU, LUG, AGO,
SET, OTT, NOV, DIC.
```

Che fa a questo punto il programmatore? Se sa già che tutto ciò che potrà richiedere al programma sarà di selezionargli le schede di certi mesi, allora leggerà semplicemente il campo dei mesi in un certo formato (BCD, per esempio) e confronterà il risultato delle letture (che chiameremo MESE) con una serie di costanti che sarà stata sua premura definire. Ne verranno statemens del tipo:

```
IF (MESE . EQ . 'MAG') GO TO ...................... ed
analoghi.
```

Se il programmatore è più smaliziato, e sa che le idee e le neccessità germogliano e crescono con l'uso stesso del programma, allora prevederà che possano emergere richieste come "separiamo le schede *dei mesi che vanno da* giugno a settembre, perchè durante i mesi estivi etc. etc." In questo caso farà bene, in fase di lettura a:

— leggere in BCD il mese,
— confrontare MESE con 'GEN', 'FEB', etc.,
— decidere di che mese si tratta,
— porre in una variabile di comodo (KMESE) 1 se si tratta di gennaio, 2 se si tratta di febbraio ...

A questo punto alla richiesta di cui sopra potrà rispondere subito con una IF

```
IF ((KMESE . GE . 6) . AND . (KMESE . LE . 9)) ...
```

Corso sul Pascal I dati in generale scalari, matrici

Essenzialmente ci siamo costruiti un *numero* (tipo sul quale *si possono* effettuare delle operazioni) da associare a 'GEN' 'FEB', etc. (su cui *non si possono* effettuare operazioni). Gli esempi che ho citato sono al solito in FORTRAN, ma le cose sono simili anche per gli altri linguaggi.

Pascal evita tutto questo giro vizioso. L'utente può introdurre un nuovo tipo che chiamerà MESE e che sarà definito solo per i valori che l'utente stesso specificherà. Su questi "valori" l'utente potrà operare senza dover ricorrere a variabili ausiliarie come detto prima.

Eccovi - al solito - il BNF

<definizione di tipo> :: = *TYPE* (4.1)
< identificatore di tipo>
= < identificatore>
{,< identificatore> };

che non è niente di nuovo rispetto a quanto visto nella Parte II. Notate ancora una volta l'uso dell'operatore di relazione = e la differenza che passa tra ":=" e "=". Più precisamente,

— scrivere := vuol dire "il valore di tutto ciò che sta a destra *viene assegnato a* ciò che sta a sinistra";
— scrivere = vuol dire una delle due cose:
  a) ciò che sta a sinistra *è la stessa cosa di* ciò che sta a destra;
  b) ciò che sta a sinistra *è definito con* ciò che sta a destra..

Torniamo ora all'esempio dei mesi. Potrete scrivere e definire un tipo di variabile detto MESE così:

```
TYPE MESE = (GEN, FEB, MAR, APR, MAG, GIU,
LUG, AGO, SEP, OTT, NOV, DIC);            (4.2)
```

Questo definisce nel vostro programma un nuovo tipo di variabile che non è *nè* intera, *nè* reale, *nè* booleana, *nè* carattere: è appunto il MESE. Questa può assumere i 12 valori che voi avete specificato e solo quelli. Così come la variabile booleana ne assume solo due: Falso e Vero. Vi faccio noare che *non* si può scrivere SET per SETTEMBRE in quanto SET è una parola riservata (metacostante).

Come si può operare in questo tipo? Sono lecite le seguenti operazioni:
— l'applicazione di tutti gli operatori relazionali
— l'uso delle funzioni

```
SUCC (X), PRED (X), ORD (X).              (4.3)
```

Gli operatori relazionali = e <> non pongono problemi. Più interessante è invece il fatto che "un mese *può esse-*

re minore di un altro". Così:

GEN < FEB < MAR < APR < .... (4.4)

In realtà - se ci pensate un attimo - ciò che succede è che Pascal fa per voi il lavoro che faceva quel tale programmatore che associava numeri a mesi: valgono insomma fra i "valori" dei mesi le stesse relazioni che intercorrono fra i numeri che ad essi venivano fatti corrispondere. Tali numeri sono dati per altro dalla funzione ORD (X) il cui uso è spiegato -al solito - molto meglio con un esempio.

```
ORD (GEN) = 0                    (4.5)
ORD (FEB) = 1
...............
ORD (DIC) = 11
```

Notate che si comincia a contare da zero. Sempre con un esempio possiamo spiegare l'uso delle altre funzioni: infatti le altre operazioni possibili sono

```
SUCC (GEN)  dà FEB               (4.6)
SUCC (MAR)  dà APR
PRED (AGO)  dà LUG
PRED (DIC)  dà NOV
```

Naturalmente il tentativo di fare

PRED (GEN) o SUCC (DIC) (4.7)

dà una segnalazione di errore che dipende dalla implementazione del sistema, ma che in sostanza non è nulla di concettualmente diverso da un *underflow* o da un *overflow*.

Vi sottolineo il fatto che non esiste l'operazione inversa di ORD: questa è un'altra delle limitazioni contro cui la gente sta protestando, e non è escluso che venga tolta in future implementazioni, visto che, dopo tutto, nel tipo CHAR esiste già l'operazione inversa di ORD. Non è però semplice eliminare questa limitazione. Se esaminate le (4.5) troverete che il compilatore "sa" di che tipo è l'argomento delle funzioni e di che tipo deve essere il risultato richiesto, ma se voleste estendere questo tipo di operazione a tipi diversi da CHAR od inventati da voi, vi trovereste subito di fronte ad una difficoltà: all'ipotetica funzione inversa di ORD in questo caso verrebbe dato come argomento un numero intero ed essa dovrebbe sapere in qualche modo qual è il tipo che volete in uscita.
Facciamo un esempio: supponiamo che abbiate definito come tipi nuovi sia i giorni della settimana (a partire da lunedì) sia le note musicali (a partire da DO). Se alla funzione ORD date come argomento "Mercoledì", questa vi restituirà il numero 3 (terzo giorno a partire da lunedì); lo stesso numero vi sarà restituito se le date come argomento la nota MI (terza a partire dal DO). Se però pensate di invertire il processo vi accorgerete che vi manca un'informazione: se all'inversa di ORD (che vi ricordo chiamarsi CHR ed essere definita solo per i caratteri) deste come argomento 3, questa non saprebbe se rispondere MI o MERCOLEDI', a meno che non specificaste (oltre a 3) anche il tipo che volete in uscita. Ciò forse verrà fatto in futuro, ma per ora non c'è questa possibilità.
Con ciò abbiamo praticamente finito con i tipi scalari speciali. Prima di passare a degli esempi concreti, forse sarà meglio vedere anche i tipi *subrange*, così da avere alla fine un'idea completa di come stanno le cose e da poter fare anche degli esempi più significativi.

I tipi subrange

Non vi spaventate, non è niente di complicato: il fatto è che spesso, mentre i valori di un tipo (nel senso generalizzato indicato prima) possono variare da un valore minimo ad un valore massimo (nel senso esteso già visto prima i "valori" possono essere mesi o colori dell'arcobaleno o note musicali, etc.), capita talvolta di avere la necessità di trattare in un modo speciale un certo insieme di valori consecutivi che per qualche ragione hanno delle proprietà speciali.
Facciamo ancora l'esempio dei mesi dell'anno. In questo caso spesso i mesi estivi vanno trattati diversamente dagli altri: sono i mesi delle vacanze in cui vuoi le fatture, vuoi i piani di lavoro, vuoi gli ordini del materiale vanno esaminati in modo diverso dagli altri. Ebbene in questo caso potrete definire un tipo *subrange* (un qualcosa come un *sottotipo*) che chiamerete "mese estivo" o magari MESEST con una dichiarazione del tipo

```
TYPE MESEST = GIU ..... AGO;          (4.8)
```

con riferimento ovviamente alla (4.2). Con questa definizione ciò che succederà sarà che potrete lavorare sui mesi GIU, LUG, AGO come se fossero un tipo a sè, ove questo si rendesse necessario.
Ci sono altre possibilità di definizione, per così dire implicite. Se noi scriviamo per esempio

```
VAR DECADE: 1 ... 10; VENTINA: 1 ... 20;    (4.9)
```

definiamo automaticamente dei tipi SUBRANGE chiamati in questo caso DECADE e VENTINA. Dal momento che il limite minimo e massimo sono interi, questi due tipi sono dei tipi subrange dei *numeri interi*.

Così, mentre sappiamo che il tipo CHAR comprende oltre alle lettere dell'alfabeto anche una serie di segni speciali, che sfortunatamente non sono uguali per tutte le implementazioni, possiamo definire un tipo subrange di CHAR che chiamiamo magari LETTER e che è costituito dalle sole lettere dell'alfabeto. Così:

```
VAR LETTER: 'A' ... 'Z';                    (4.10)
```

oppure

```
TYPE LETTER: 'A' ... 'Z';                   (4.11)
```

Le due definizioni sono del tutto equivalenti e, incidentalmente, sono molto comuni in tutti i programmi data la particolare importanza che rivestono le lettere dell'alfabeto entro il tipo CHAR. Non fosse altro che per ricerche di archivio, ordinamento di dati in ordine alfabetico e simili.

Introduzione ai dati strutturati

È noto che con i dati organizzati singolarmente (gli scalari, come finora li abbiamo chiamati) si può fare ancora ben poco. Lo scopo stesso del calcolo automatico è quello di effettuare delle iterazioni, cioè di ripetere più volte, magari in condizioni diverse, dei calcoli più o meno complessi: ciò richiede automaticamente che i dati possano venire organizzati in qualche modo in memoria e che si possa aver accesso non solo al dato singolo ma ad una collettività di dati.
A queste esigenze provvedono appunto i dati strutturati, i quali - è bene dirlo - sono anch'essi dei tipi particolari e come tali vanno regolarmente definiti all'inizio del blocco in cui serviranno.
Le strutture che Pascal permette sono parecchie (molte di più che non nel FORTRAN) e sono precisamente le seguenti:

array, record, set, file

con l'aggiunta di un tipo per così dire ausiliario o collaterale (il *pointer*) che vedremo più avanti.
Va subito detto che lo schema a blocchi, o il nesting, se volete, che è una regola generale in Pascal, si applica senz'altro a tutti i dati strutturati: potrete quindi avere *arrays di records, files di arrays, records di arrays di files* etc. essendo le uniche limitazioni dovute al sistema implementato, al buon senso e all'efficienza. Infatti è bene notare che non è sempre ovvio il modo migliore e più efficiente per organizzare i dati: questa è veramente un'arte che richiede soprattutto pratica, oltrechè intuizione, idee chiare, ed estro.

Una breve nota: il tipo *set* fa eccezione in quanto può essere costituito solo da elementi scalari o subrange. Non è permesso, per esempio, un *set di set*, o un *set di array*.

Il tipo array

Cominciamo quindi con un tipo *array*. Questo è costituito da un numero *fisso* di *elementi* o *componenti* (elements, components) tutti *di uno stesso tipo*, chiamato anche il *tipo dei componenti* o il *tipo di base* (component type, base type). Il numero dei componenti deve essere dichiarato e non può venire mutato: anche questa è una limitazione che si pensa verrà travolta dallo sdegno popolare, essendo molto comodo poter disporre di arrays con un numero di componenti che viene specificato non quando si scrive il programma, ma quando esso viene eseguito. Comunque staremo a vedere.
Come può essere individuato un componente? Nel modo più ovvio: anzitutto dicendo a quale array appartiene, e poi specificando il posto che esso occupa in questo array. L'informazione del posto occupato è quindi data da una variabile detta *indice* (index) che può essere di qualunque tipo, non necessariamente un intero.
Vi faccio notare (per coloro che credono di sapere già queste cose) che un array *non è* una matrice, o meglio, non necessariamente. Una matrice è una struttura di *numeri* (reali od interi a scelta) nella quale uno o più indici individuano il singolo elemento. E tali indici *sono interi* (magari possono essere anche negativi). Un array invece è ad *un solo indice* (simile eventualmente a ciò che si dice matrice unidimensionale o vettore) costituito da elementi di tipo *qualunque* con l'indice di tipo qualunque (escluso quello reale). Questa distinzione è molto importante per evitare confusioni presenti o future.
Con il procedimento di nesting si possono però costruire arrays di arrays: queste strutture sono (con le limitazioni dette prima circa i tipi dei componenti e degli indici) proprio le usuali matrici che si incontrano in matematica e che si scrivono per esempio

$$\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix} \qquad (4.12)$$

etc. Va anche qui però fatta un pò d'attenzione. Anzitutto la rappresentazione matriciale tipo la (4.12) va bene solo per uno o due indici: se uno volesse rappresentare una matrice a tre indici dovrebbe uscire dal foglio, e con quattro indici anche dal nostro Universo fisico. Poi la

SIEMENS

per continuare a servirvi sempre meglio, la ed elettromeccanici - ha creato una rete di concessionari autorizzati, questi:

Ge.P. Elettronica
via Quintino Sella, 32 bis
tel. (049) 66.45.00 - 66.43.06
35100 Padova

STOREL s.r.l.
via Lombardi, 49
tel. (051) 37.28.23 - 37.20.80
telex DUEBI i 211670
40128 Bologna

carter s.r.l.
via delle Cascine, 22/24
tel. (055) 36.44.12 - 36.84.87
50144 Firenze

COMMIND s.r.l.
via Chiusure, 329/A
tel. (030) 31.84.36
telex 0494
25100 Brescia

siamo a vostra completa disposizione... e n

SIEMENS ELETTRA S.P.A.

Corso sul Pascal

(4.12) porta fuori strada il lettore, in quanto non gli dice nulla riguardo al modo in cui i dati sono memorizzati nel calcolatore. Più utile è pensare ad un'array come ad una pila di piatti: un piatto si individua dicendo di che pila si tratta e poi dicendo che posto occupa questo piatto nella pila di piatti. Un piatto allora verrà individuato:
a) dal nome della "pila di pile"
b) dal posto che vi occupa la pila in cui c'è il piatto
c) dal posto che il piatto occupa nella sua pila
Il BNF è il seguente:

```
<tipo>::=< tipo semplice >                    (4.13)
         < tipo strutturato >
         < tipo pointer >
<tipo semplice>::=< tipo scalare >
                  < tipo subrange >
                  < identificatore di tipo >
<tipo array>::=ARRAY [ < tipo dell'indice >
               {, < tipo array > } ]
               OF < tipo dei componenti >;
```

oppure

```
<tipo array>::=ARRAY [ < tipo dell'indice > ]
               {OF ARRAY[< tipo dell'indice>]}
                OF < tipo dei componenti >
                                              (4.14)
```

Vedete subito che la notazione (4.13) corrisponde ad indicare un array in forma matriciale, mentre la seconda ad indicarlo come "pila di pile di piatti". Inutile dire che la prima forma è di gran lunga la preferita per la sua compatezza.
Così, per esempio, se vogliamo definire con MATR X1 una matrice "classica" di numeri reali, con il primo indice che varia da 1 a 10 ed il secondo da 1 a 7 scriveremo

```
VAR MATR X1 : [ 1 ... 10, 1 ... 7 ] OF REAL;(4.15)
```

e potremo anche avere una matrice di variabili booleane (false o vere essendo FALSO < VERO) a tre indici che variano il primo tra 1 e 10, il secondo tra -2 e 9, ed il terzo tra -40 e -10:

```
VAR MATR X2: ARRAY [ 1 .. 10, -2 .. 9, -40 .. -10 ]
             OF BOOLEAN;                  (4.16)
```

Se abbiamo definito il tipo MESE come nella (4.2), potremo definire una matrice di variabili booleane che ci dica in quale mese viene tenuto acceso il riscaldamento ed in quale no:

```
VAR RISCALD: ARRAY [ MESE ] OF BOOLEAN; (4.17)
```

Se supponiamo poi che un certo signore fa il lavativo e, caso strano, è malato sempre in giorni da lunedì a venerdì, definiremo i seguenti tipi:

```
TYPE GIORNO   = (LUN, MAR, MER, GIO,
                 VEN, SAB, DOM);      (4.18)
     GIORLAV  = (LUN ..... VEN);
     GIORFEST = (SAB ..... DOM);
```

e potremo avere le seguenti matrici:

```
VAR MALATO   : ARRAY [ GIORNO ]       (4.19)
               OF BOOLEAN;
    LAVATIVO : ARRAY [ GIORLAV ]
               OF BOOLEAN;
    JELLATO  : ARRAY [ GIORFEST ]
               OF BOOLEAN;
```

E potremmo fare di peggio, esaminando il suddetto signore nei vari mesi dell'anno con una matrice a due indici:

```
VAR ASTUZIA : ARRAY [ MESE, GIORLAV ]
              OF BOOLEAN;             (4.20)
    JELLA   : ARRAY [ MESE, GIORFEST ]
              OF BOOLEAN;
```

Le possibilità di inventare tipi, di avere dei tipi subrange e di costruire arrays con indici e variabili qualsiasi dà in mano al programmatore un attrezzo veramente poderoso. Quando poi avremo finito di esaminare tutti i tipi, vedrete che spesso sarete imbarazzati di fronte all'enorme flessibilità e potenza di questa parte del linguaggio. Prima di vedere alcuni esempi, devo parlarvi di un'altra possibilità: può essere necessario a volte (anche a costo di allungare i tempi di esecuzione del programma e di appesantire oltre misura il compilatore) risparmiare spazio in memoria. A ciò provvede l'operazione di impacaggio (packing).
Questo è un tipo di operazione non facile da definire in generale, dato che abbiamo visto che i tipi possono essere "di fantasia", e che la loro rappresentazione all'interno della memoria dipende dall'implementazione del sistema. Un caso particolare sono però i caratteri, i quali occupano di solito un byte in una parola: riservare loro una parola mentre in 16 bits ne occupano la metà non solo è uno spreco in memoria, ma spesso occorre averli "di fila" in memoria, per poterli magari scrivere. È ciò che si chiama - se ricordate - *una stringa* di caratteri. La stringa è quindi definita proprio come un "array impaccato". Per esempio

```
VAR STRINGA: PACKED ARRAY [ 1...37] (4.21)
             OF CHAR;
```

è una stringa di 37 caratteri.
Se però voleste avere accesso ad uno solo di questi caratteri, la cosa non sarebbe più così semplice; l'operazione di impaccaggio fa sì che in una sola parola del calcolatore vengano messe parecchie variabili e allora delle due l'una: o aggredite il problema a livello di assembler o usate le funzioni standard di Pascal. Queste sono:

```
PACK (LARGE, K, SMALL)          (4.22)
UNPACK (SMALL, LARGE, K)
```

dove con LARGE e SMALL si sono indicate le matrici "grande" (non impaccata) e "piccola" (impaccata).
Per vedere cosa queste funzioni vogliono dire vediamo un caso: supponiamo di essere in presenza di un mini da 16 bits e di una matrice di caratteri. Poichè ogni carattere occupa un byte (8 bits), se noi non diciamo nulla, il compilatore metterà un carattere in ogni parola, che risulterà pertanto "mezza vuota". È possibile però salvare spazio con un'operazione di "packing": se la matrice LARGE (i cui caratteri sono memorizzati uno per parola) ha 100 elementi, e la matrice SMALL 50, l'operazione di packing metterà due elementi di LARGE in un elemento di SMALL permettendo un notevole risparmio di memoria.
Come avvenga esattamente il processo non interessa chi programma: l'importante è che PACK e UNPACK siano operazioni l'una inversa dell'altra.
Supponiamo quindi che sia:

```
SMALL: PACKED ARRAY [ U ... V ] OF CHAR;(4.23)
LARGE: ARRAY [ M ... N ] OF CHAR;
```

L'operazione

```
PACK (LARGE, K, SMALL)          (4.24)
```

metterà (V-U) elementi di LARGE in SMALL a partire dall'elemento K incluso, con l'opportuna convenzione di impaccaggio decisa in fase di progettazione. Se ripetiamo questo per tutti i K (M <= K <= U - V + N) alla fine in SMALL avremo la matrice LARGE impaccata.
Questa peraltro non è un'operazione frequente. Più spesso può avvenire di dover affrontare il problema opposto, di dover cioè riferirsi ad un elemento di una matrice impaccata. Tipico il caso del carattere di una stringa. A questo provvede la UNPACK

```
UNPACK (SMALL, LARGE, K)          (4.25)
```

che metterà in LARGE (K), LARGE (K + 1), ..., LARGE (K + V - U) l'elemento K di SMALL "disimpaccato". Continuando il processo per tutti i K in LARGE si troverà la serie di singoli valori di SMALL: nel nostro caso ci sarà un carattere per parola. In ogni caso l'ordine in cui appariranno dipende dal particolare sistema usato.
È bene notare una cosa: cercate - se lo potete - di evitare l'uso di queste procedure in quanto sono molto dispendiose in tempo e in memoria.
Possono essere indispensabili, è vero, ma meno si usano e meglio è.

Un esempio

Ed eccovi a raccogliere i frutti di questa tirata un pò astratta con un esempio realistico.
Anzitutto ritorniamo al concetto di array: oltre all'analogia dei piatti ne esiste un'altra che forse rende meglio conto del fatto che, mentre gli *elementi* di un array devono essere sempre dello stesso tipo, il tipo degli indici può essere arbitrario. Si tratta dell'analogia della biblioteca: consideriamo una biblioteca di scaffali dove ci sono dei libri (sempre lo stesso numero per scaffale) con uguale numero di pagine, ognuna delle quali ha lo stesso numero di righe, ognuna delle quali ha lo stesso numero di caratteri. Come facciamo ad individuare *una lettera* nella biblioteca? Dovremo dare un nome alla biblioteca, poi dare il numero dello scaffale (primo indice), poi specificare il libro sullo scaffale (secondo indice), la pagina del libro (terzo indice), la riga della pagina (quarto indice), il carattere della riga (quinto indice). Ciò è esattamente ciò che succede quando parliamo di un array o matrice a cinque indici.
Notate un fatto importante: *il tempo* che spendiamo per cercare una certa lettera non dipende dalla sua posizione nella bibblioteca; sia che si tratti della prima, sia che si tratti dell'ultima lettera della biblioteca, il tempo di ricerca è lo stesso. È per questo che un array (o matrice) si dice anche *accesso casuale* (random access). È un'espressione brutta ed infelice che può portare fuori strada, però vuol dire appunto questo.
Supponiamo ora di avere una matrice MATR1 (ad una dimensione) che contenga dei caratteri, e di voler costruire una seconda matrice MATR2 in cui gli stessi caratteri siano messi in ordine alfabetico. Supponiamo che la matrice MATR1 consista di 100 elementi. Seguiremo un procedimento molto semplice e *non* ottimizzato (ci sono procedimenti più furbi di questo!): esamineremo il primo carattere con il secondo, il primo con il terzo, ..., il secondo con il terzo, il secondo con il quarto, ..., ed ogni volta che troviamo che non viene rispettato l'ordine alfabetico, scambieremo i due elementi tra loro. Continueremo questi passaggi finchè non ci saranno più inversioni: a questo punto ci fermeremo.

Ecco il programma:

```
PROGRAM ALFORD (OUTPUT);
TYPE LETTER : 'A' ... 'Z';
VAR  MATR1  : ARRAY [ 1 ... 100 ] OF LETTER;
     MATR2  : ARRAY [ 1 ... 100 ] OF LETTER;
     J, K, INVERS : INTEGER;
FOR  J := 1 TO 100 DO
     MATR2 (J) := MATR1 (J);
REPEAT  { START LOOP }
     INVERS := 0;
     FOR J := 1 TO 99 DO
     FOR K := J+1 TO 100 DO
       BEGIN
         IF MATR2 (J) > MATR2 (K) THEN
           BEGIN TEMP := MATR2 (K);
                 MATR2 (K) := MATR1 (J);
                 MATR2 (J) := TEMP;
                 INVERS := INVERS +1;
           END
       END;
UNTIL INVERS > 0
END
```

Tre osservazioni:

— Quando c'è un nesting di FOR come in questo caso, la seconda FOR non ha bisogno di una parentesi logica "BEGIN" ..... "END". Infatti essa, con gli statements che seguono (questi sì chiusi in parentesi logica) è *un unico* statement che pertanto può venir messo subito dopo la parola DO.

— Capita spesso di avere diverse END una di seguito all'altra. Non c'è modo di evitarle, nè si prevede di farlo in futuro. È un'altra di quelle costrizioni volute che servono a mantenere i programmi ordinati.

— In questo programma c'è un errore. Quale? ■

software per Apple II

ple
pte

Via Bartolomeo della Gatta 26/28 - tel 055/713369 - 50143 Firenze

SOFTWARE PER GESTIONE AZIENDALE

GESTIONE MAGAZZINO — L. 360.000 + IVA
Un programma professionale per il Vs. Apple. La riduzione delle scorte e la statistica sulla movimentazione resa possibile da una efficiente gestione del magazzino, ripaga largamente da sola l'acquisto di un Apple. Richiede 48K di memoria, da 1 a 4 dischi, eventuale stampante.

FATTURAZIONE E BOLLETTAZIONE — L. 360.000 + IVA
Automatizzate l'emissione di bolle e fatture con Apple. Il Programma attinge informazioni dall'archivio Clienti e dall'archivio di magazzino ed è collegato alla successiva contabilità IVA. Richiede 48K di memoria da 1 a 4 dischi, Stampante.

CONTABILITÀ CLIENTI — L. 420.000 + IVA
Risolvete finalmente con Apple il problema di una gestione efficiente delle Vendite. Automaticamente estratti conto, giornali bollati IVA, controllo del Credito, analisi del Fatturato, etc.

CONTABILITÀ FORNITORI — L. 420.000 + IVA
Forse più delle vendite Vi preme gestire bene i Vs. acquisti. Le stesse prestazioni della contabilità Clienti con 2 giornali IVA. (disponibile da giugno 80)

CONTABILITÀ GENERALE — L. 420.000 + IVA
L'ovvio completamento di una contabilità sezionale unito ai precedenti. Utilizzato da solo, se non avete problemi di IVA, Vi fornirà sempre un'aggiornata situazione del piano dei conti e, cosa più importante, un giornale fiscale e la situazione del conto "profitti e perdite". (disponibile da Maggio 80)

AUTOLISTINO — L. 280.000 + IVA
Un listino-Prezzi Intelligente destinato ad imprese commerciali di vendita all'ingrosso e al dettaglio. Attinge informazioni dall'archivio magazzino, incrementa o decrementa il prezzo di gruppi di articoli o intere categorie. Ricava il prezzo vendita in base ad un'aliquota prefissata. Calcola automaticamente gli sconti per quantitativi.

PAGHE E STIPENDI — L. 360.000 + IVA
Un programma professionale e generalizzato che soddisfa le esigenze di 15 diversi contratti Nazionali di Lavoro.
Particolarmente interessante per i Consulenti.

PAGHE EDILI — L. 360.000 + IVA
Una versione speciale del programma PAGHE E STIPENDI per gli operai del settore Edilizio.

DISTRIBUZIONE GIORNALI — L. 500.000 + IVA
Risolvere tutti i problemi di gestione delle Agenzie di distribuzione Giornali alle Edicole, compresa la gestione delle spedizioni, gestione resi, emissione di bolle e fatture.
Prevede fino a 600 Testate in distribuzione fino a 125 Edicole.

ANALMEDIC 1 — L.450.000 + IVA
Programma per Laboratori di Analisi Mediche: Gestione completa del Libro di ingressi o ACCETTAZIONE. Prepara automaticamente il giornale delle analisi da effettuare dopo i prelievi giornalieri, stampa le etichette per le provette, e i moduli e/o veline dei risultati (per l'archivio) come da Normativa Vigente.
È composto da un menù principale e da 12 menù secondari, suddivisi per tipo di analisi.
Richiede 48K, 2 dischi, Stampante.

ANALMEDIC 2 — L. 320.000 + IVA
Programma per Laboratori di Analisi Mediche: Gestione completa dell'archivio clienti, registra tutte le analisi fatte e loro data, stampa la scheda dell'analisi con i risultati, è legato al LIBRO INGRESSI del precedente ANALMEDIC 1.
Richede 48K, 2 dischi, Stampante.

AMMINISTRAZIONE CONDOMINI — L. 360.000 + IVA
Programma completo per la gestione condominiale, calcola automaticamente le quote suddivise in millesimi voce per voce, gestisce la posizione dei condomini per acconti, saldi, sospesi. Effettua la stampa dettagliata del giornale di gestione e degli estratti conto.
Richiede 48K, 1 o 2 dischi, Stampante.

COMMERCIO E DISTRIBUZIONE 1 — L. 450.000 + IVA
Un Programma destinato ad imprese commerciali all'ingrosso e/o dettaglio: attinge i dati dall'archivio creato col programma MAGAZZINO, da la possibilità dell'articolo/i richiesto/i, calcola l'IVA, stampa un giornale delle movimentazioni effettuate (numerate conindice azzerabile e/o impostabile), crea un archivio movimentazioni, effettua la statistica di tutte le operazioni eseguite, scarica automaticamente le merci vendute dall'archivio MAGAZZINO.
Richiede 48K, almeno 2 dischi, Stampante.

COMMERCIO E DISTRIBUZIONE 2 — L.450.000 + IVA
Simile al precedente, con lo scarico e la stampa dettagliata suddivisa in più reparti dell'azienda, analoga suddivisione per la parte statistica.
Richiede 48K, almeno 2 dischi, Stampante.

GESTIONE ALBERGHIERA COMPUTERZZATA

FRONT OFFICE PER ALBERGHI — L. 400.000 + IVA
La soluzione ideale per Alberghi di qualsiasi dimensione, dalle 50 camere in sù. Gestione delle Prenotazioni nel tempo (fino ad un anno). Gestione degli arrivi con quadratura contabile con la saldaconti. Gestione statistica (Presenze) ed economica (media per presenze). Gestione vendita Saloni.

BACK OFFICE PER ALBERGHI — (in preparazione)
AMMINISTRAZIONE PER ALBERGHI — (in preparazione)
GESTIONE RISTORANTE, BAR, MAGAZZINO — (in preparazione)

VISICALC — L. 175.000 + IVA
Un potente ed originalissimo strumento di archiviazione e di calcolo. Trasforma Apple in una grande matrice di dati legati fra loro da relazioni semplici o complesse.
Cambiando un dato vengono automaticamente ricalcolati quelli che da esso dipendono.
Ideale per la computazione di calcoli contributivi, scala mobile, Listini prezzi con molte scale sconti, etc.
Richiede 48K, 1 disco, Stampante.

ALLESTIMENTO E GESTIONE ARCHIVI

DATA BASE MANAGEMENT SYSTEM 1 — L. 250.000 + IVA
Uno strumento di grande generalità. Consente di sviluppare immediatamente la maggior parte delle piccole e medie applicazioni gestionali di Apple, senza scrivere una riga di Programmi.
Fra le caratteristiche principali: introduzione, aggiornamento di dati, ordinamenti (fino a 20) anche multichiave, stampe personalizzate, ricerche selettive, totalizzazioni. Richiede 48K di memoria, 2 dischi, Stampante.

MINI DATA BASE — L. 110.000 + IVA
Una versione ridotta del Programma precedente. Richiede 48K di memoria, 1 disco, eventuale Stampante.

DATA BASE MANAGEMENT SYSTEM 2 — L. 360.000 + IVA
Una versione più estesa del System 1, ideale per usi anagrafici, Statistici, Pubblicitari indirizzati a specifici settori merceologici, Ordinamenti e ricerche per Laboratori di analisi Mediche, ed altre vastissime applicazioni. Richiede 48K di memoria, almeno 2 dischi, Stampante.

WHATSIT - DATA BASE A CAMPI LIBERI — L. 175.000 + IVA
Un originalissimo archivio a campi liberi, si possono cioè diminuire od aumentare i dati di un nominativo senza essere legati ad uno schema predeterminato.
Si interroga la Macchina come interrogare una Persona: es. "quando l'appuntamento con l'ing. ROSSI?" - risposta "l'appuntamento con l'ing. ROSSI è alle 10 del 23/5/80
Il programma è in lingua inglese, ma i dati possono essere inseriti in Italiano. Richiede 48K di memoria, 1 disco, linguaggio Integer Basic

ELABORAZIONE DI TESTI - WORD PROCESSING -

APPLE WRITER — L. 79.000 + IVA
Trasformate il Vostro Apple in un potente sistema di scrittura (word Processor). Collegando ad Apple una stampante a margherita (per un carattere perfetto) oppure una stampante veloce (Rosy 26) potrete produrre con enorme efficienza testi ripetitivi, contratti, offerte, circolari, etc. Richiede 48K, 1 disco, 1 Stampante

EASYWRITER — L. 95.000 + IVA
Trattasi di un "word processor" molto sofisticato, con particolari istruzioni per l'incolonnamento, modifica, cancellazione, introduzione di nuove righe di testo, etc. Richiede 48K di memoria, 1 disco, 1 stampante.

APPLEPOST — L. 79.000 + IVA
Se doveste frequentemente spedire materiale pubblicitario o di documentazione a centinaia di indirizzi, Applepost risolverà brillantemente il problema di arichiviare, aggiornare, stampare Etichette.
Richede 48K, 1 disco, 1 stampante.

PROGRAMMI DI UTILITÀ

MICROMEMO — L. 79.000 + IVA
Il Vostro Apple si trasforma con questo programma in una potente agenda computerizzata, ricorda appuntamenti, scadenze, pagamenti, note personali. Per un intero anno i Vs. impegni personali o di lavoro vengonomemorizzati da questo programma, permettendoVi una rapida consultazione. Richiede 48K, 1 disco.

RICOPIA DISCHETTI CON SINGOLO DRIVER — L. 150.000 + IVA
Se volete "mettere al sicuro" i Vs. programmi e quindi il Vs. lavoro, pur possedendo un Apple con 1 solo disk-driver, potrete ricopiare i Vs. dischetti con questo utilissimo programma, che non è indispensabile per il Vs. lavoro.
Richiede 48K di memoria, 1 disco.

RUBRICA TELEFONICA INTELLIGENTE — L. 30.000 + IVA
La ricerca di un nome o di un numero ricordandosi soltanto parzialmente uno di essi vene effettuata in pochi istanti da questo interessante programma. È un potente aiuto per la Vs. memoria, dato che per la ricerca basta una sola sillaba, al resto ci pensa il Vs. Apple.
Contiene 150 numeri (può essere estesa sino a 1000). Nella versione minima richede 16K, × disco.

SOME COMMON BASIC PROGRAMS — L. 50.000 + IVA
Trattasi di un set di programmi per calcoli finanziari, satistica, diagnostica, miscellaneus (varie). Richiede 48K, 1 disco

CONTRIBUTOR'S PROFRAMS VOL. 1/2 — L.30.000 + IVA
Programmi applicativi per i campi più svariati. Sono di grande aiuto al programmatore in Basic Applesoft. Richiede 48K, 1 disco.

CONTRIBUTOR'S PROGRAMS VOL. 3/4/5 — L. 45.000 + IVA
Logico seguito al n 1 ed al numero 2. richiede 48K, 1 disco

APPLESCACCHI MICROCHESS — L. 30.000 + IVA
È molto difficile battere il Computer, lui non ha ne sviste ne indecisioni: comunque è possibile, prevedendo almeno 80 mosse.

FUNCTION PLOT HI RESOLUTION — L. 50.000 + IVA
Come realizzare molte funzioni grafiche con Apple. 48K, 1 disco

CAPABILITIES DEMO DISKETTE — L. 50.000 + IVA
Dimostrativo delle capacità grafiche di Apple. 48K, 1 disco.

DIMOSTRATIVO DI INGEGNERIA CIVILE — L. 50.000 + IVA
In PASCAL UCDS - Disponibile da Giugno 80.

PROGRAMMA DI DUPLICAZIONE DISCHETTI — L. 35.000 + IVA
In Italiano. richiede 48K, 2 Dischi.

I ns. pakages sono duplicabili con i 2 programmi di duplicazione in elenco

Un cross-disassemblatore (quasi) intelligente

di **S. Margarita**

Come decifrare i programmi oggetto del microprocessore 8080 e svelarne i misteri.

Introduzione

La programmazione degli elaboratori elettronici (micro e non) ha ormai raggiunto un grado di semplificazione tale da poter essere affidata in molti casi agli utenti stessi. Tale progresso, che contribuisce notevolmente allo sviluppo dell'informatica "per tutti", è dovuto principalmente all'evoluzione dei linguaggi di programmazione, la cui struttura tende ad avvicinarsi sempre di più al linguaggio umano. Inoltre tali linguaggi, in buona parte slegati dalla struttura fisica della macchina, rivolgendosi più all'aspetto applicativo del problema che all'aspetto tecnico, richiedono poche conoscenze di elaborazione dati da parte dell'utilizzatore.
Il collegamento tra questi linguaggi (Pascal PL/1 - BASIC - COBOL...), la cui struttura è oramai molto lontana dalla macchina intelligibile dal calcolatore rimasto immutato viene svolto tramite compilatori e interpreti.
Mentre dal lato applicativo gli strumenti a disposizione degli utenti sono sempre più potenti e sofisticati, dal lato tecnico invece, per quanto riguarda cioè i programmi di base, non si è assistito ad un'evoluzione così rapida ed importante. È vero che certi linguaggi evoluti permettono di accedere a risorse interne della macchina tali da consentire la realizzazione addirittura di programmi di base come sistemi operativi, compilatori, supervisori per telecomunicazioni; però il linguaggio più usato per tali applicazioni rimane l'Assembler.
Tale linguaggio, e questo ne spiega la complessività, è molto vicino al linguaggio macchina: ad ogni istruzione in linguaggio macchina corrisponde una istruzione in Assembler, che ne è soltanto la rappresentazione mnemonica.
Sono gli *assemblatori* ad operare la traduzione delle istruzioni Assembler in linguaggio macchina. Un assemblatore ha quindi come ingresso un programma sorgente, composto da un insieme di istruzioni scritte in linguaggio Assembler, e come uscita un programma oggetto equivalente, in binario, eseguibile dalla macchina.

I disassemblatori

Un disassemblatore è invece un programma che effettua le operazioni inverse: ammette in ingresso un programma oggetto e ne restituisce la traduzione in linguaggio Assembler.
In pratica quindi traduce un insieme scarsamente intelligibile di cifre binarie (o esadecimale, a seconda della rappresentazione) in una serie di istruzioni molto più chiare e di facile utilizzazione da parte del programmatore. Il disassemblatore non costituisce un vero e proprio strumento di aiuto alla programmazione, bensì un mezzo per rendere più facile la manutenzione dei programmi già scritti, o meglio per interpretare i programmi scritti da altri e già memorizzati sui supporti più diversi: periferiche tradizionali (dischi, nastri magnetici, nastri perforati, schede perforate,...) e perfino su dispositivi interni (memoria come PROM, ROM,....). uesto evidentemente può essere fatto per gli scopi più diversi: modificare o migliorare programmi esistenti oppure tradurli in un Assembler eseguibile su un'altra macchina, operazione impossibile partendo dal programma oggetto. L'obiettivo della presente realizzazione è pertanto di fornire uno strumento realmente operativo che consenta di raggiungere tali scopi.
È tenendo presente questo obiettivo che si è deciso di realizzare un cross-disassemblator e e non semplicemente un disassemblatore.
Ricordiamo che per cross-disassemblatore si intende un programma che disassembla i moduli oggetto di un elaboratore e che ha bisogno di un elaboratore diverso per essere eseguito. Lo stesso vale per i cross-compilatori o i cross-assemblatori che sono, per esempio disponibili, per la maggior parte dei microprocessori, sugli elaboratori PDP/11 e DEC-10. In questo modo, pur dovendo sopportare alcuni lievi inconvenienti, come la conversione dei supporti per esempio, si evitano i limiti insiti nella configurazione del microelaboratore usufruendo di tutte le risorse di un elaboratore, e fornendo in tal modo al programmatore uno strumento veramente potente.
Per la realizzazione è stato utilizzato il PL/1, questo programma richiede un elaboratore IBM/370 con sistema operativo O.S. Si rimanda al paragrafo intitolato *Nuova versione* per la problematica legata alla conversione di linguaggio o al cambiamento di elaboratore.
Per prima cosa esaminiamo le istruzioni del microprocessore 8080.

Struttura delle istruzioni del microprocessore 8080

Non esporremo qui in dettaglio l'insieme delle istruzioni dell'8080, ma ne esamineremo semplicemente le caratteristiche principali, quelle cioè che assumono particolare importanza in questa realizzazione.

Lunghezza delle istruzioni: le istruzioni dell'8080 possono avere una lunghezza pari a 1, 2 o 3 bytes (insiemi di 8 bits). Le istruzioni di 1 byte sono composte soltanto dal codice operativo (es.: POP, PUSH,...). Le istruzioni di due bytes contengono un primo byte di codice operativo e un secondo byte di dato o di indirizzo (es.: MVI, OUT,...). Infine le istruzioni di 3 bytes sono composte sempre dal codice operativo nel primo byte, ed inoltre da un indirizzo o un dato (di 16 bits) nel secondo e terzo byte a questo proposito bisogna precisare che nel secondo byte viene memorizzata la parte bassa del dato o dell'indirizzo mentre la parte alta (cioè quella più significativa) è memorizzata nel terzo byte.
Modi di indirizzamento: l'8080 dispone di 4 modi di indirizzamento:
— diretto: il secondo e il terzo byte dell'istruzione contengono l'indirizzo del dato;
— da registro: l'istruzione indi-

ca il registro o la coppia di registri che contiene il dato;
— da registro indiretto: l'istruzione indica una coppia di registri che contiene l'indirizzo del dato;
— immediato: l'istruzione contiene il dato stesso e può avere una lunghezza di 1 o 2 bytes.

Trasferimento dell'esecuzione: a meno di interruzioni esterne o di istruzioni di trasferimento, lo svolgimento del programma avviene sequenzialmente, un'istruzione dopo l'altra. Una istruzione di trasferimento può specificare l'indirizzo della prossima istruzione da eseguire in 2 modi:
— diretto; l'istruzione stessa contiene l'indirizzo dell'istruzione da eseguire (in genere nel byte 2 e nel byte 3);
— indiretto; l'istruzione di trasferimento fa riferimento ad una coppia di registri che contiene l'indirizzo dell'istruzione da eseguire.

Per i nostri scopi sarà utile adottare la seguente partizione dell'insieme delle istruzioni dell'8080 (tra parentesi viene indicato il numero di istruzioni di ogni sottoinsieme):

1) Restart (8): i cui codici operativi sono C7, CF, D7, DF, E7, EF, F7, FF (in esadecimale come le successive indicazioni se non diversamente precisato);
2) Jump incondizionato (1): codice operativo C3;
3) Jump condizionato (8): codici operativi C2, CA, D2, DA, E2, EA, F2, FA;
4) Call incondizionato (1): codice operativo CD;
5) Call condizionato (8): codici operativi C4, CC, D4, DC, E4, EC, F4, FC;
6) Return incondizionato (1): codice operativo C9;
7) Return condizionato (8): codici operativi C0, C8, D0, D8, E0, E8, F0, F8;
8) Jump HL indiretto (1): codice operativo E9;
9) Altre istruzioni che non influenzano la sequenza esecutiva del programma (208).

Il massimo numero di istruzioni diverse rappresentabili con un codice operativo di 8 bits è pari a 256. Nel caso dell'8080 soltanto 244 istruzioni esistono effettivamente mentre 12 configurazioni di 8 bits non vengono utilizzate.
Come si vedrà in seguito queste 12 configurazioni saranno trattate separatamente.

Struttura dei moduli oggetto

Per capire i problemi di realizzazione di un disassemblatore esaminiamo la struttura generale di un programma oggetto. Può essere rappresentato come un alternarsi di aree contenenti istruzioni (codici operativi e operandi) di aree contenenti dati (dati veri e propri o aree di lavoro usate dal programma).
Si considera che il punto di entrata del programma, come in genere avviene, coincida con l'inizio del programma: pertanto il programma oggetto inizierà con un'area di istruzioni mentre successivamente si incontreranno aree di dati (*segmento D*) e aree di istruzioni (*segmento I*) in alternanza (v. figura 1).
Se il programma oggetto fosse, come a volte accade, composto da un unico segmento *I*, disassemblarlo sarebbe banale.
Per disassemblare un segmento *I* è sufficiente infatti:
— costruire una tabella (dizionario) che ad ogni codice operativo associ la lunghezza in bytes e il mnemonico dell'istruzione;
— procedere sequenzialmente nell'esame del segmento *I*, considerando che il primo byte è un codice operativo, e nella ricerca e decodifica dei codici operativi e degli operandi in funzione della lunghezza dell'istruzione;
— stampare, o registrare su altro supporto evidentemente, il mnemonico corrispondente ricavato dal dizionario.

Se il programma, per ipotesi, fosse composto da un unico segmento *D*, l'operazione sarebbe ancora più semplice (anche se qui non si tratterebbe realmente di disassemblare), occorendo:
— costruire un dizionario che ad ogni carattere, come viene rappresentato sul supporto sul quale risiede il programma da disassemblare (binario, caratteri esadecimali, formato Intel...), associ la rappresentazione esterna dei caratteri; soprattutto se prima della stampa si richiede una conversione dei caratteri (ASCII, EBCDIC....); del segmento *D*, secondo le indicazioni del dizionario;
— convertire sequenzialmente i caratteri;
— stampare o registrare i caratteri ottenuti.

Da questo breve esame della struttura dei moduli oggetto si può desumere che la principale difficoltà nel disassemblare un programma oggetto sta nel riuscire a distinguere i segmenti *I*, e i segmenti *D*; eseguita questa operazione, infatti, basterà applicare le due metodologie esposte in precedenza, ognuna al segmento corrispondente.
Definiamo inoltre sottosegmento (*I* o *D*) un sottoinsieme qualsiasi di un segmento *I* o *D*.
Queste considerazioni e l'analisi delle istruzioni del microprocessore 8080 ci permettono di trarre alcune conclusioni che ci saranno di aiuto nella fase di realizzazione vera e propria del disassemblatore:
— l'indirizzo contenuto in una istruzione di salto o di salto condizionato è quello di un segmento *I* o di un sottosegmento *I* (l'esecuzione del programma non può passare ad un'area di dati);
— l'indirizzo contenuto in una istruzione di chiamata di sottoprogramma (chiamata condizionata o no) è quello di un segmento o di un sottosegmento *I*;
— dopo un'istruzione di trasferimento condizionato (salto o chiamata di sottoprogramma) prosegue il segmento *I* e sono quindi presenti istruzioni (che infatti verranno eseguite se non è verificata la condizione);
— dopo un'istruzione di salto incondizionato può sia proseguire il segmento *I* in cor-

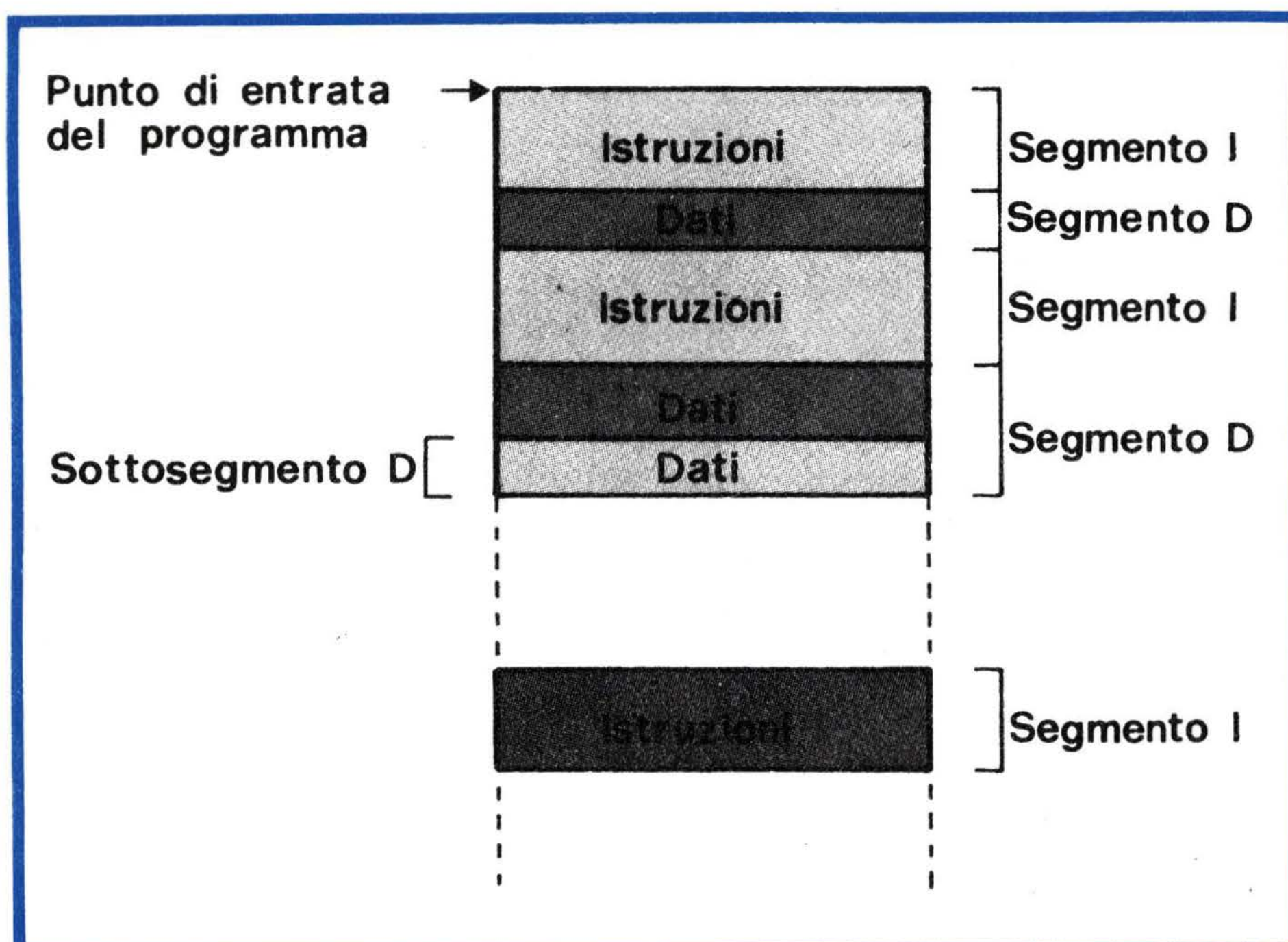



Figura 1 - Struttura tipica di un programma oggetto.

so sia iniziare un segmento *D*;
— dopo un'istruzione di chiamata incondizionata vi possono essere dati (in genere argomenti passati al sottoprogramma chiamato) sia istruzioni, eseguite al ritorno dal sottoprogramma;
— le istruzioni di Restart rimandano all'indirizzo di un segmento o di un sottosegmento *I*.

Criteri di realizzazione

Diversi metodi possono essere adottati per risolvere il problema della separazione dei segmenti *I* e dei segmenti *D*.
Quello che è stato adottato nella presente realizzazione consiste nel simulare l'esecuzione vera e propria del programma, seguendo il flusso logico delle istruzioni.
Questo ci è sembrato il metodo più rigoroso e logico in quanto, in fase di esecuzione, il programma distingue evidentemente i dati delle istruzioni!
I problemi che si manifestano a questo punto derivano dalla struttura non sequenziale del flusso delle istruzioni eseguite; lo stesso programma può infatti differire da una esecuzione all'altra a seconda del verificarsi in certe condizioni che ne determinano lo svolgimento. Vedremo ora come vengono affrontati tali problemi.

Struttura del disassemblatore

Il disassemblatore consiste in un programma particolarmente strutturato, suddiviso in 4 blocchi, ognuno corrispondente ad una particolare fase di esecuzione:
1) *inizializzazione*
2) *lettura*
3) *identificazione*
4) *traduzione*

esaminiamo in dettaglio ognuna di queste quattro fasi.

La fase di inizializzazione

Consiste nella lettura delle opzioni e nella lettura del dizionario. Per quanto concerne la lettura delle opzioni, è disponibile una serie di opzioni che permettono di facilitare e migliorare l'utilizzazione del programma. Le principali sono:
— SIZE, che specifica l'ampiezza del programma da disassemblare, per ottimizzare, tramite allocazione dinamica, lo spazio occupato in memoria (il valore di difetto è di 4 kbytes);
— INIT, che specifica l'indirizzo di partenza (di caricamento) del programma da

Un cross-disassemblatore

disassemblare (il valore di difetto è 0);
- DESC, indica se si vuole o no la descrizione di ogni istruzione;
- MNEM, che permette di scegliere tra la stampa dei mnemonici dell'8080 e di quelli dello Z80 (microprocessore compatibile) per eventuali esigenze di conversione;
- SKIP, PAGS, BIT7, che facilitano la gestione della stampa e la lettura dei dati.

Per quanto concerne il dizionario, questo è esteso ai 256 codici operativi possibili, e la sua struttura viene esemplificata nel Listato 1.
Esso contiene, per ognuna delle 256 voci, il codice operativo, la rappresentazione stampabile, se esiste, del carattere ASCII corrispondente il mnemonico 8080, il mnemonico Z80, la lunghezza in bytes dell'istruzione e la descrizione dell'operazione effettuata.
La possibilità di adottare i mnemonici dello Z80 costituisce un buon aiuto, come abbiamo già accennato, alla conversione dei programmi dell'8080 allo Z80, mentre la descrizione può aiutare alla comprensione del programma, una volta disassemblato.
Il codice operativo e la lunghezza invece vengono usati dal disassemblatore. Il messaggio "Initializathion phase terminated" indica la conclusione di questa prima fase.

La fase di lettura

Consiste nelle operazioni seguenti:
- lettura del programma da disassemblare, previa scelta, mediante opzione, del formato da utilizzare (binario, esadecimale, Intel....) e nella memorizzazione in formato binario di tali dati;
- controllo della coerenza dei dati letti con il formato di rappresentazione (se per esempio il formato è quello esadecimale, si controlla che siano presenti soltanto caratteri dallo 0 al 9 e dalla A alla F).

Questa fase si conclude con il messaggio "Reading phase terminated". Se invece lo spazio occupato dal programma è superiore a quello allocato inizialmente, l'esecuzione del programma viene interrotta e il messaggio "Allocate more space for data" viene inviato.

La fase di identificazione

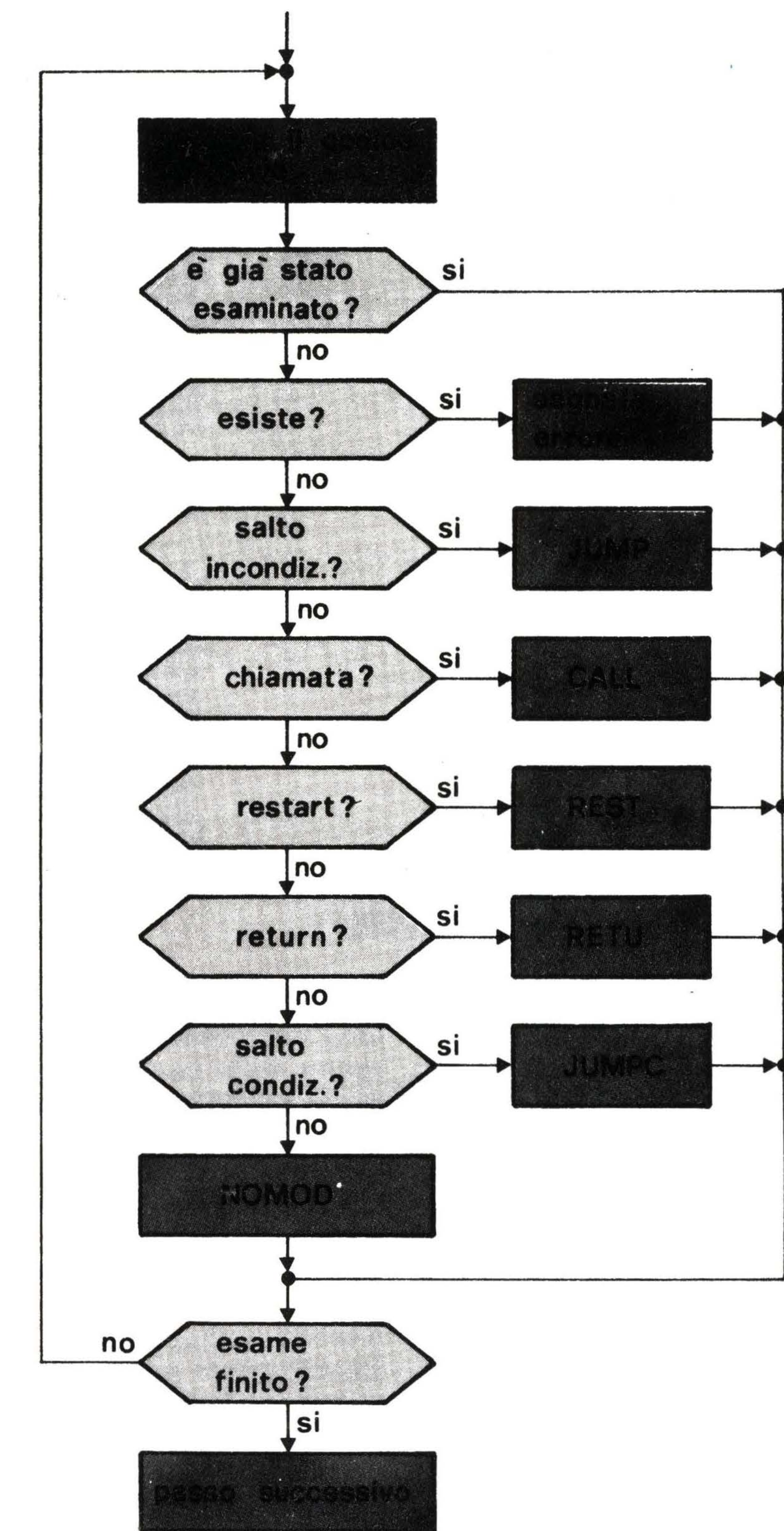
È la fase più complessa del programma. È quella che identifica i diversi segmenti(*D* o *I*) del programma oggetto. Per meglio capire la sua struttura e il suo funzionamento esaminiamo la procedura più importante che viene usata nei suoi vari passi: la procedura START. Tale procedura innanzitutto controlla la validità del codice operativo. Se questo non esiste, lo segnala con un messaggio, se invece esiste chiama la procedura relativa al codice operativo incontrato. Le procedure impiegate sono le seguenti:
- JUMP per il salto incondizionato;
- CALL per le chiamate condizionate e non;
- REST per i restart;
- RETU per i return;
- JUMPC per i salti condizionati;
- NOMOD per tutte le altre istruzioni.

In questa versione del disassemblatore tutte le chiamate vengono trattate nello stesso modo, ed inoltre i ritorni condizionati vengono trattati come istruzioni generiche. Vedremo in seguito come viene gestita l'istruzione PCHL (codice operativo E9).
La presente procedura viene eseguita per tutti i codici operativi, seguendo il flusso logico delle istruzioni, e questo soltanto se il codice operativo in esame non è già stato esaminato in precedenza. Per questo, ad ogni byte del programma viene associato un indicatore. Inizialmente tutti gli indicatori assumono il valore *D* (il programma oggetto viene cioè considerato come un unico segmento *D*); assumono il valore *I* (Istruzioni) quando il byte corrispondente viene riconosciuto come un codice operativo e come tale trattato dalla relativa procedura.
Illustriamo come viene seguito il flusso logico del programma oggetto esaminando le funzioni delle 7 procedure elencate.
Tutte svolgono 3 funzioni comuni:
- ricercano nel dizionario la lunghezza dell'istruzione;
- assegnano agli indicatori

```
00      NOP              NOP            1 NO OPERATION
01      LD BC,DDDDH      LXI B,DDDDH    3 LOAD REGISTER PAIR IMMEDIATE
02      LD (BC),A        STAX B         1 STORE ACCUMULATOR INDIRECT
03      INC BC           INX B          1 INCREMENT REGISTER PAIR
04      INC B            INR B          1 INCREMENT REGISTER
05      DEC B            DCR B          1 DECREMENT REGISTER
06      LD B,DDH         MVI B,DDH      2 MOVE IMMEDIATE
07      RLCA             RLC            1 ROTATE LEFT
08      EX AF,AF'                       0
09      ADD HL,BC        DAD B          1 ADD REGISTER PAIR TO H AND L
0A      LD A,(BC)        LDAX B         1 LOAD ACCUMULATOR INDIRECT
0B      DEC BC           DCX B          1 DECREMENT REGISTER PAIR
0C      INC C            INR C          1 INCREMENT REGISTER
0D      DEC C            DCR C          1 DECREMENT REGISTER
0E      LD C,DDH         MVI C,DDH      2 MOVE IMMEDIATE
0F      RRCA             RRC            1 ROTATE RIGHT
10      DJNZ DISP                       0
11      LD DE,DDDDH      LXI D,DDDDH    3 LOAD REGISTER PAIR IMMEDIATE
12      LD (DE),A        STAX D         1 STORE ACCUMULATOR INDIRECT
13      INC DE           INX D          1 INCREMENT REGISTER PAIR
14      INC D            INR D          1 INCREMENT REGISTER
15      DEC D            DCR D          1 DECREMENT REGISTER
16      LD D,DDH         MVI D,DDH      2 MOVE IMMEDIATE
17      RLA              RAL            1 ROTATE LEFT THROUGH CARRY
18      JR DISP                         0
19      ADD HL,DE        DAD D          1 ADD REGISTER PAIR TO H AND L
1A      LD A,(DE)        LDAX D         1 LOAD ACCUMULATOR INDIRECT
1B      DEC DE           DCX D          1 DECREMENT REGISTER PAIR
1C      INC E            INR E          1 INCREMENT REGISTER
1D      DEC E            DCR E          1 DECREMENT REGISTER
1E      LD E,DDH         MVI E,DDH      2 MOVE IMMEDIATE
1F      RRA              RAR            1 ROTATE RIGHT THROUGH CARRY
20      JR NZ,DISP                      0
21      LD HL,DDDDH      LXI H,DDDDH    3 LOAD REGISTER PAIR IMMEDIATE
22 "    LD (ADDRH),HL    SHLD ADDRH     3 STORE H AND L DIRECT
23 #    INC HL           INX H          1 INCREMENT REGISTER PAIR
24 $    INC H            INR H          1 INCREMENT REGISTER
25 %    DEC H            DCR H          1 DECREMENT REGISTER
26 &    LD H,DDH         MVI H,DDH      2 MOVE IMMEDIATE
27 /    DAA              DAA            1 DECIMAL ADJUST ACCUMULATOR
28 (    JR Z,DISP                       0
29 )    ADD HL,HL        DAD H          1 ADD REGISTER PAIR TO H AND L
2A *    LD HL,(ADDRH)    LHLD ADDRH     3 LOAD H AND L DIRECT
2B +    DEC HL           DCX H          1 DECREMENT REGISTER PAIR
2C '    INC L            INR L          1 INCREMENT REGISTER
2D -    DEC L            DCR L          1 DECREMENT REGISTER
2E .    LD L,DDH         MVI L,DDH      2 MOVE IMMEDIATE
2F /    CPL              CMA            1 COMPLEMENT ACCUMULATOR
30 0    JR NC,DISP                      0
31 1    LD SP,DDDDH      LXI SP,DDDDH   3 LOAD REGISTER PAIR IMMEDIATE
32 2    LD (ADDRH),A     STA ADDRH      3 STORE ACCUMULATOR DIRECT
33 3    INC SP           INX SP         1 INCREMENT REGISTER PAIR
34 4    INC (HL)         INR M          1 INCREMENT MEMORY
35 5    DEC (HL)         DCR M          1 DECREMENT MEMORY
36 6    LD (HL),DDH      MVI M,DDH      2 MOVE TO MEMORY IMMEDIATE
37 7    SCF              STC            1 SET CARRY
38 8    JR C,DISP                       0
39 9    ADD HL,SP        DAD SP         1 ADD REGISTER PAIR TO H AND L
3A :    LD A,(ADDRH)     LDA ADDRH      3 LOAD ACCUMULATOR DIRECT
3B ;    DEC SP           DCX SP         1 DECREMENT REGISTER PAIR
```

Listato 1 - Struttura del dizionario.

relativi ai bytes dell'istruzione il valore *I* che segnala che questa è già stata disassemblata;
- incrementano il puntatore all'istruzione successiva da disassemblare di un valore pari alla lunghezza dell'istruzione appena esaminata se un'istruzione che non modifica il flusso logico del programma; invece assegnano al puntatore il valore dell'indirizzo della nuova istruzione, se modifica il flusso del programma;

Inoltre ognuna svolge delle funzioni specifiche:
- NOMOD: riconosce l'istruzione PCHL;
- JUMPC: questa procedura crea una tabella, con organizzazione LIFO, (*Last In First Out*) nella quale va inserito l'indirizzo ricavato dall'istruzione di salto condizionato, se non è già presente in tabella. Questi indirizzi infatti saranno degli indirizzi di segmenti o sottosegmenti di tipo *I*, e verranno usati da altre procedure;
- JUMP: dai bytes 2 e 3 calcola l'indirizzo di salto (se questo è inferiore all'indirizzo di inizio del programma o superiore a quello di fine, stampa un messaggio); se il byte corrispondente a questo indirizzo è stato in precedenza disassemblato, reperisce nella tabella dei salti condizionati l'ultimo indirizzo che non è ancora stato esaminato e lo assegna al puntatore della prossima istruzione da disassemblare; se la tabella è vuota, se cioé non ci sono più indirizzi di salti condizionati da esplorare, lo segnala come una probabile incongruenza del programma da disassemblare (probabile loop);
- CALL: viene ricavato l'indirizzo del sottoprogramma chiamato; se tale indirizzo è fuori dal programma da disassemblare, viene ignorato (caso, per esempio di chiamata di sottoprogrammi di un altro programma). Viene creata e aggiornata una tabella LIFO del punto di ritorno dopo l'esecuzione del sottoprogramma, cioé dell'indirizzo dell'istruzione successiva a quella di chiamata. Al puntatore di istruzione viene assegnato l'indirizzo del sottoprogramma chiamato, a condizione che questo non sia già stato disassemblato, nel qual caso viene disassemblata l'istruzione successiva a quella di chiamata;
- REST: questa procedura agisce come la precedente; l'unica differenza sta nel calcolo dell'indirizzo di destinazione che in questo caso va calcolato utilizzando 3 bits dell'istruzione di restart;
- RETU: assegna al puntatore di istruzione l'indirizzo dell'ultimo punto di ritorno introdotto in tabella dalla procedura CALL, segnalando se questa tabella è vuota. Si può notare come questa tabella sia pressoché equivalente allo stack del microprocessore.

Alla procedura START che chiama queste 6 procedure viene fornito semplicemente il valore del puntatore di istruzione; questa identifica il codice operativo e passa il controllo alla procedura interessata. Dopo aver svolto le proprie funzioni detta procedura ripassa il controllo alla procedura START, fornendole un nuovo valore del puntatore, ottenuto nei modi che abbiamo appena visto.
Tale funzionamento è rappresentato dal diagramma di Figura 2.
La fase di identificazione è composta di 3 passi:

- una prima simulazione dell'esecuzione delle istruzioni del programma da disassemblare, secondo le modalità appena viste, cioé tramite la procedura START. Questo passo permette di identificare un primo insieme di segmenti *I*. In genere dopo l'esecuzione di questo passo il programma non è interamente disassemblato, però la tabella degli indirizzi dei salti condizionati in questo caso risulta non vuota; la funzione del secondo passo è proprio l'esame di questa tabella;
- il secondo passo, partendo da ogni indirizzo contenuto nella tabella (che, come abbiamo visto, è sicuramente l'indirizzo di un segmento o di un sottosegmento *I*) ripe-

è già stato esaminato? — si / no
esiste? — si / no
salto incondiz.? — si: JUMP / no
chiamata? — si: CALL / no
restart? — si: REST / no
return? — si: RETU / no
salto condiz.? — si: JUMPC / no
NOMOD
esame finito? — no / si
passo successivo

Figura 2 - Diagramma di flusso della procedura START.

```
PROGRAM SIZE IS 08674 BYTES (DEC.)
                          INTELLIGENT CROSS-DISASSEMBLER ICD8080 VERSION 2.3                PAGE    1

HEX.    DEC.    ISN.    CODE               STATEMENT                DESCRIPTION

0020    00032   00001   31AA1C    RST4     LXI SP,1CAAH             LOAD REGISTER PAIR IMMEDIATE ]  Procedura PROG
0023    00035   00002   C3541B             JMP 1B54H                JUMP                         ]
0026    00038   00003                      DB  000A0A0A0A0A38303830204D445320H        8080 MDS  ]
0036    00054   00004                      DB  4D4143524F20415353454D424C455220H  MACRO ASSEMBLER ]  Procedura DATA
0046    00070   00005                      DB  56455253494F4E20312E300909095041H  VERSION 1.0   PA ]
0056    00086   00006                      DB  4745200D0A0A0A090900H              GE        ]
0060    00096   00007   C303F8             JMP F803H                JUMP                         ]
0063    00099   00008   C9                 RET                      RETURN

0064    00100   00009   C306F8             JMP F806H                JUMP
0067    00103   00010   C9                 RET                      RETURN

0068    00104   00011   21341E             LXI H,1E34H              LOAD REGISTER PAIR IMMEDIATE
006     00107   00012   71                 MOV M,C                  MOVE TO MEMORY
006C    00108   00013   C309F8             JMP F809H                JUMP
006F    00111   00014   C9                 RET                      RETURN

0070    00112   00015   21351E             LXI H,1E35H              LOAD REGISTER PAIR IMMEDIATE
0073    00115   00016   71                 MOV M,C                  MOVE TO MEMORY
0074    00116   00017   C30CF8             JMP F80CH                JUMP
0077    00119   00018   C9                 RET                      RETURN

0078    00120   00019   21361E             LXI H,1E36H              LOAD REGISTER PAIR IMMEDIATE
007B    00123   00020   71                 MOV M,C                  MOVE TO MEMORY
007C    00124   00021   C30FF8             JMP F80FH                JUMP                            Procedura PROG
007F    00127   00022   C9                 RET                      RETURN

0080    00128   00023   C315F8             JMP F815H                JUMP
0083    00131   00024   C9                 RET                      RETURN

0084    00132   00025   C318F8             JMP F818H                JUMP
0087    00135   00026   C9                 RET                      RETURN

0088    00136   00027   21371E             LXI H,1E37H              LOAD REGISTER PAIR IMMEDIATE
008B    00139   00028   3600               MVI M,00H                MOVE TO MEMORY IMMEDIATE
008D    00141   00029   3E77               MVI A,77H                MOVE IMMEDIATE
008F    00143   00030   21371E             LXI H,1E37H              LOAD REGISTER PAIR IMMEDIATE
0092    00146   00031   96                 SUB M                    SUBTRACT MEMORY
0093    00147   00032   DAA200             JC 00A2H                 JUMP ON CARRY(CY=1)
0096    00150   00033   0E00               MVI C,00H                MOVE IMMEDIATE
0098    00152   00034   CD7000             CALL 0070H               CALL
009B    00155   00035   21371E             LXI H,1E37H              LOAD REGISTER PAIR IMMEDIATE
009E    00158   00036   34                 INR M                    INCREMENT MEMORY
009F    00159   00037   C28D00             JNZ 008DH                JUMP ON NOT ZERO(Z=0)
00A2    00162   00038   C9                 RET                      RETURN                       ]
```

Listato 2 - Programma disassemblato. (Macro Assembler dell'MSD Intel).

te il procedimento di simulazione dell'esecuzione delle istruzioni con l'uso della procedura START. Il programma cioé disassembla nuovamente a partire da ognuno degli indirizzi contenuti nella tabella. A questo punto tutto o quasi tutto il programma oggetto è disassemblato. Può però presentarsi la necessità di forzare il disassemblatore su certi indirizzi, per 2 motivi: se dopo una prima passata all'esame del tabulato ci si accorge che non tutto il programma è stato disassemblato e si riescono ad individuare dei segmenti *D* che in realtà sono segmenti *I* e devono ancora essere disassemblati: oppure se, come accade spesso, il programma contiene una tabella di indirizzi ad uso dell'istruzione PCHL; in questo caso tali indirizzi sono sicuramente indirizzi di segmenti o di sottosegmenti *I*;

— perciò il terzo passo accetta 2 tipi di dati in ingresso, che memorizza: indirizzi di partenza di zone da disassemblare oppure indirizzi di partenza e di fine tabelle di indirizzi. Gli indirizzi che vanno forniti sono facilmente ricavabili dal tabulato ottenuto da una prima passata del programma. Infine questo terzo passo disassembla, nello stesso modo dei passi precedenti, a partire da detti indirizzi.

Il disassemblatore, abbiamo detto, esegue il programma oggetto secondo la sequenza logica delle istruzioni, però in pratica, come abbiamo visto, esegue effettivamente soltanto le istruzioni di trasferimento; la versione attuale non esegue le istruzioni che non modificano lo svolgimento sequenziale del programma.
Questo significa che quando incontra un'istruzione PCHL, che vincola al contenuto dei registri H e L l'indirizzo di salto, l'indirizzo di destinazione non è noto, non essendo noto il contenuto della coppia di registri HL. In questo caso il disassemblatore interrompe il passo in esecuzione e trasmette il controllo al passo successivo. Se stava già eseguendo il terzo viene conclusa la fase di identificazione.
Alla fine dei 3 passi della fase di identificazione (conclusa con il messaggio "Identification phase terminated") il programma oggetto è disassemblato. L'indicatore associato ad ogni byte assume cioè il valore *D* se è un byte di dati, e il valore *I* se è il codice operativo e l'operando di un'istruzione. A questo punto subentra la quarta ed ultima fase, quella di traduzione.

La fase di traduzione

Non esamina più il programma oggetto ma soltanto gli indicatori ad esso associati. Questi indicatori, uno per ogni byte come abbiamo visto, possono assumere 2 valori: il valore *D* per i bytes di dati e il valore *I* per i bytes di istruzioni. L'esame di questi indicatori avviene sequenzialmente. Se il valore dell'indicatore è *I* viene chiamata la procedura PROG, se invece è *D* viene chiamata la procedura DATA. Vediamo quali sono le funzioni di queste due procedure. La procedura PROG gestisce la decodifica e la stampa delle istruzioni del programma disassemblato. Viene riportato nel Listato 2 un esempio della stampa ottenuta da questa procedura.
L'intestazione di ogni pagina è costituita dal numero progressivo e dal nome del programma. Le informazioni disponibili al programmatore, o all'utente in genere, sono per ogni istruzione: l'indirizzo esadecimale (HEX.), l'indirizzo decimale (DEC.), il numero progressivo dell'istruzione (ISN.) la rappre-

Un cross-disassemblatore

sentazione esadecimale dell'istruzione (CODE), il mnemonico dell'istruzione (STATEMENT) e infine la descrizione della funzione (DESCRIPTION).
Per maggior chiarezza i vari sottoprogrammi vengono separati in fase di stampa, ed inoltre nel mnemonico delle istruzioni di 3 bytes viene ristabilito l'ordine logico degli indirizzi o dei dati (parte più significativa a sinistra), pur non essendo stata sentita la necessità di convertire gli indirizzi esadecimali in etichette. La descrizione - opzionale - usata è quella dell'Intel 8080 Microcomputer Systems User's Manual. Infine, se gli indirizzi di Restart vengono utilizzati dal programma, prima del mnemonico viene inserita l'etichetta relativa: RST0, RST1, RST2 ....
La procedura DATA invece gestisce la decodifica e la stampa delle aree dei dati. Dallo stesso Listato 2 si ha un esempio di stampa di questa procedura. Si notano informazioni comuni anche alla procedura PROG, come gli indirizzi esadecimale e decimale. La numerazione progressiva è relativa qui ad un'insieme di al massimo 16 bytes, mentre non viene riportata la configurazione esadecimale nella colonna CODE in quanto presente nel corso stesso dell'istruzione, preceduta da DB (Define Byte). Infine viene riportata la decodifica corrispondente, soltanto evidentemente per i caratteri stampabili.

Limitazioni

Da questo rapido esame della struttura e del funzionamento di questo disassemblatore sono facilmente individuabili le principali limitazioni:

— segue il flusso logico del programma soltanto considerando le istruzioni di trasferimento che contengono esplicitamente l'indirizzo di destinazione (non è però un grosso inconveniente per il microprocessore 8080 che dispone soltanto di una istruzione il cui indirizzo di trasferimento non è identificabile immediatamente - PCHL -);
— questa versione è in grado di disassemblare soltanto dei programmi oggetto per il microprocessore 8080. Anche se tale microprocessore è molto diffuso, è abbastanza vecchio ed è stato superato da microprocessori più avanzati, compatibili (Z80) o non.
— non è interattivo, e pertanto possono essere neccessarie più passate, ognuna con delle indicazioni supplementari fornite dall'utente per arrivare ad avere il programma oggetto completamente disassemblato.

Nuova versione

Per ovviare ai primi due ordini di inconvenienti è in fase di realizzazione una nuova versione che disponga delle seguenti funzioni:

— simulazione dell'esecuzione non soltanto delle istruzioni di trasferimento, ma anche delle istruzioni che non modificano il corso del programma. Questo per disporre, in ogni momento di questa esecuzione simulata, del contenuto dei registri al fine di risolvere almeno in parte il problema delle istruzioni come PCHL, presenti in numero abbastanza elevato nell'insieme di istruzioni dello Z80 per esempio;
— estensione del programma disassemblatore all'insieme di istruzioni dello Z80 (quello dell'8080 come è noto ne é un sottoinsieme);
— è inoltre allo studio di includere in questa nuova versione la possibilità di rendere il programma disassemblatore indipendente dal microprocessore per il quale è stato scritto il programma. Si tratta in pratica di generalizzare il programma in modo da permettergli di disassemblare qualsiasi programma oggetto, fornendogli semplicemente un dizionario leggermente più complesso di quello usato nella presente realizzazione.

Per quanto riguarda la possibilità di adattare questo programma ad un'altro elaboratore o di riscriverlo in un altro linguaggio si possono fare le seguenti considerazioni:

— partendo dal listato del programma in PL/1 è molto facile riscriverlo in un linguaggio a struttura simile (Pascal, PL/M, PL/Z...), mentre è più complessa la traduzione in Basic, FORTRAN, anche se realizzabile;
— lo spazio di memoria occupato dal programma attuale è di circa 48 K (questa cifra spaventerà gli utenti di microelaboratori mentre farà appena sorridere chi usa il PL/1).
Questo programma, tradotto in uno dei linguaggi evoluti per microelaboratori che abbiamo appena visto, è però eseguibile in una memoria di 20-22 K su un sistema dotato di un dischetto (Intellec MDS per esempio);
— tale spazio può essere ridotto od aumentato (con conseguente aumento o diminuzione del tempo di esecuzione) a seconda delle esigenze o delle disponibilità di memoria e di unità periferiche;
— ovviamente, se viene realizzato su un microelaboratore dotato di 8080, non rimarrà un cross-disassemblatore ma diventerà un disassemblatore.

Conclusione

Nel disassemblare un programma oggetto si può incorrere in 2 tipi di errore: disassemblare dei dati come istruzione (errore di 1° tipo). Oppure disassemblare delle istruzioni come dati (errore di 2° tipo). Dal punto di vista operativo, dal punto di vista cioé dell'utente del disassemblatore, è molto grave un errore di 1° tipo, difficilmente correggibile, che compromette la intelligibilità del programma e ne sfalsa la struttura, mentre è molto meno grave un errore di 2° tipo, che equivale semplicemente ad avere un programma solo parzialmente disassemblato, e quindi da completare in successive passate.
Il criterio di scelta dell'algoritmo utilizzato da questo disassemblatore, quello della simulazione dell'esecuzione delle istruzioni, è stato proprio quello di ottenere un programma esente da possibilità di errore di 1° tipo.
Questo vuol dire che, in un numero limitato di passate, si arriva ad un programma oggetto completamente e correttamente disassemblato. Altri algoritmi invece, di più facile realizzazione, presentano l'inconveniente di poter generare errori di 1° tipo.
Evidentemente questo disassemblatore, anche oltre i miglioramenti della prossima versione, può sicuramente essere reso più potente e più completo con l'aggiunta di nuove funzioni, dettate dalle esigenze di ogni singolo utente. Lo scopo della presente realizzazione era, ricordiamolo, di fornire all'utente uno strumento sicuro e di facile utilizzo in diverse applicazioni, senza ambizioni di sofisticazione.

I lettori interessati a questo programma possono rivolgersi direttamente all'autore al seguente indirizzo:
Dott. Sergio Margarita - Corso Vittorio Emanuele II, 209 - 10139 Torino. ■

Comparazioni logiche per localizzazione errori.

Analisi in tempo reale per ricerca guasti.

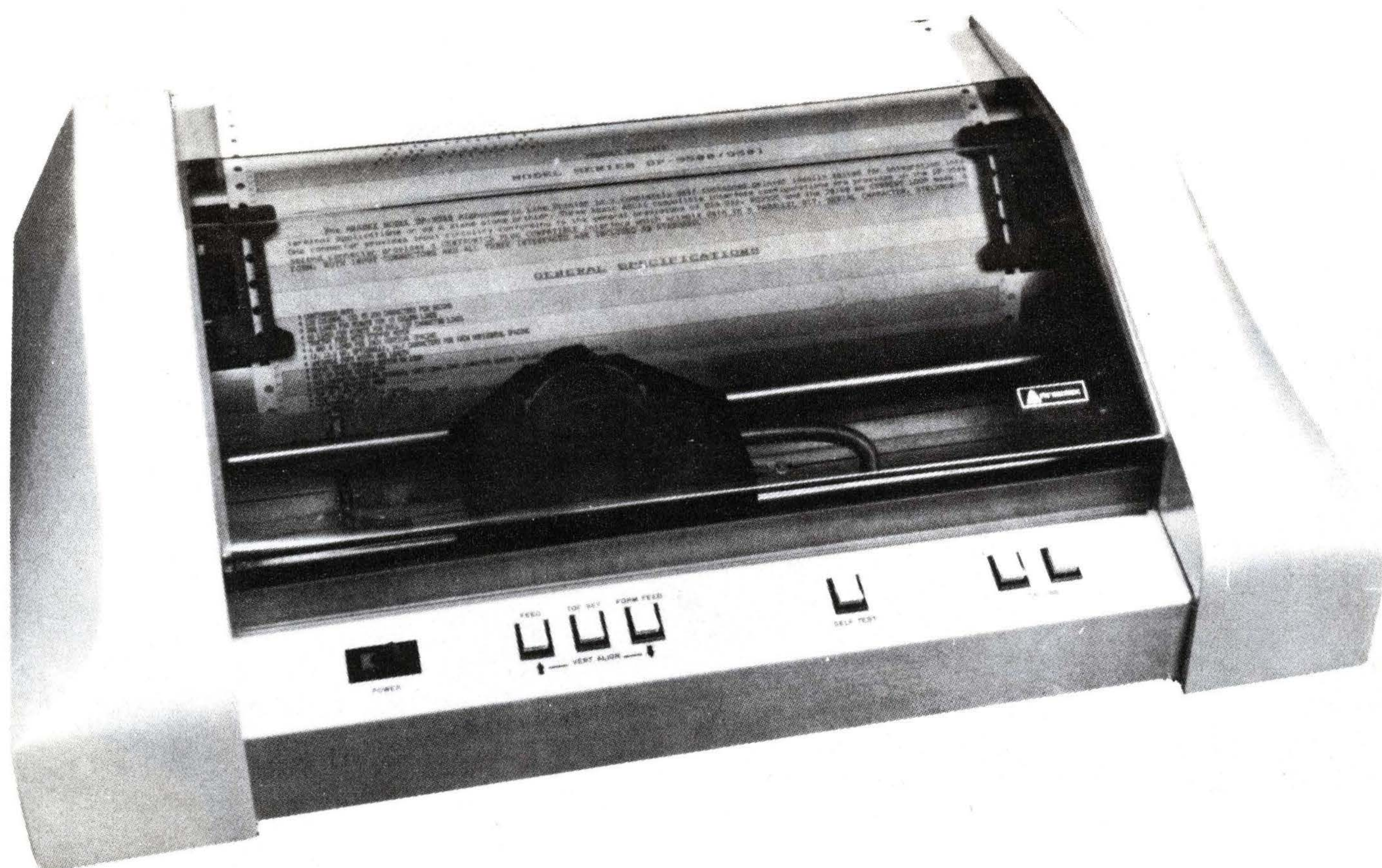
Manutenzione, test o ricerca, lo strumento è uno solo.

L'esigenza di analizzare la funzionalità logica di dispositivi elettronici non cessa allorquando i «digits» sono, per così dire, usciti dalle mani dei progettisti. Come per i volt e per gli ampère, la «logica» segue il prodotto attraverso lo sviluppo ed il collaudo e via via durante la manutenzione ed il «service»: aree ben al di fuori della fabbrica, ove la ricerca di eventuali anomalie di funzionamento della logica dei circuiti hanno sempre costituito un arduo problema.
Così è stato sin ora ed è perciò che abbiamo progettato un nuovo strumento, il **Logicoscopio PM3540** destinato a soddisfare tutte queste esigenze dal laboratorio al servizio esterno di manutenzione.
Il PM3540 è il solo strumento singolo portatile in grado di fornire sia la rappresentazione delle tabelle di dati sia la loro rappresentazione in funzione del tempo. Premendo un pulsante il PM3540 **si trasforma da logic analiyzer in un oscilloscopio** doppia traccia, con triggering eccitato dalla stessa parola logica, e con **esatto** inizio dell'oscillogramma al bit iniziale della parola di trigger (requisito essenziale per ricerca guasti digitali). Si possono: visualizzare i dati in codice binario, ottale o esadecimale: catturare blocchi da 64 x 16 bits ovunque lungo tutto il flusso dei dati e scorrerli pagina dopo pagina col solo tocco di un pulsante; scoprire la posizione ed analizzare le cause di eventuali errori; utilizzare lo strumento come un normale oscilloscopio.
La versatilità del PM3540, equipaggiato con una dotazione di sonde logiche dedicate, può essere evidenziata da una lunga serie di significanti prerogative in aggiunta ad un'eccellente funzionalità quale oscilloscopio a doppia traccia.
Letteralmente meglio delle **due funzioni analisi in tempo reale ed analisi degli stati logici** con un solo strumento, compatto, portatile e di alta classe.

Logicoscopio PM3540.

PHILIPS

PHILIPS

Philips S.p.A. - Divisione Scienza & Industria
V.le Elvezia, 2 - 20052 Monza - Tel. (039) 36.35.248
Filiali: BOLOGNA (051) 493.046
CAGLIARI (070) 666.740 **PADOVA** (049) 657.700
ROMA (06) 342.041 **TORINO** (011) 2.164.121

Cosa è successo a BIT '80?

di **Aldo Cavalcoli**

La recente mostra sui microprocessori, Home e Personale Computer, tenutasi all'US Trade Center nei giorni dall'8 all'11 maggio, sponsorizzata da BIT, ha registrato un'interessante affluenza di pubblico.
Cosa ha significato questa mostra? Per chi vive il settore dall'interno, come sua realta quotidiana, direi tecnicamente poco, se non una serie di conferme sui prodotti e sulle tendenze. Determinante invece la "fauna" dei visitatori, che con le loro caratteristiche culturali e provenienze professionali hanno permesso una valutazione reale della dimensione del fenomeno microcomputer e personal computer.
Mai come quest'anno il pubblico è stato eterogeneo, sia come attività professionale che come fasce di età.
Moltissimi i giovani, i ragazzi, sia di istituti tecnici, in bande vocianti, che in più modesti gruppi di studenti universitari, soprattutto del Politecnico.
Se appena gli lasciavate un sistema in mano, subito si mettevano a digitare, e spesso con cognizione, tipo quello studente che ha passato quasi un'ora sul PET per fare, improvvisando, un gioco televisivo, o l'altro che con noncuranza si è fatto, in linguaggio macchina, in piedi, una routine di moltiplicazione per Z80, che è girata al primo colpo.
E poi ancora, domande pertinenti, significative di un rapporto di estremo interesse con queste macchine, quasi di attaccamento, alla ricerca di verifiche di idee ed elucubrazioni personali spesso note solo teoricamente, data la cronica carenza di strumentazione.
Le generazioni più anziane erano invece caratterizzate da un fattore comune: angoscia. Gliela si leggeva negli occhi, nell'atteggiamento timido ed ansioso con cui si accostavano alle macchine, con la paura del tempo che passa, del poco tempo da dedicare per impossessarsi di tutto quello che era loro presentato, con la paura del "non ce la farò mai".
Interessantissima l'evoluzione della domanda classica: da "Faccio questo mestiere, mi occupo di questo, come posso usare i microprocomputer?" a "Dato che faccio questo lavoro, con i microcomputer vorrei fare questo e questo; è possibile?" Eccezionale! Ma allora la divulgazione sta avendo i suoi effetti, non è poi così difficile accostarsi all'intelligenza a basso costo del microcomputer.
Si è anche evidenziata la realizzazione di "isole di cultura", centrate sui grossi centri industriali, in cui i potenziali utenti non elettronici possono venire in possesso di documentazione, possono eseguire prove, realizzare uno scambio di idee ed esperienze.
Al di fuori di questo, il nulla è ancora più nulla.
Se prima c'era poco dappertutto, ed allora ci si consolava, ora si soffre per essere troppo indietro in alcune zone d'Italia. Concludendo, sarebbe bastato guardare i vari gestori di computer shop, due anni fa timorosi, pur nella loro fiducia nel futuro, ed ora sereni e soddisfatti, per dire: "È andata!".
Ma da parte mia ho gradito tutta quella serie di timide ed inarticolate domande di insospettata provenienza, rivoltemi dai vari visitatori, per confermare la correttezza di certe iniziative e l'esigenza di altre, ieri solo pensate, oggi da realizzarsi.

Musica Elettronica e Microcomputer: una proposta di applicazione real-time con il SYM-1

di **Paolo Bozzola**
Computerjob Elettronica, Brescia

Parte I

Premessa

Si è tenuto nel mese di aprile, a Milano il convegno della *FAST (Federazione delle Associazioni Scientifico Tecniche)* sul tema "La Musica e l'Elaboratore Elettronico".
Al convegno sono giunte numerose delegazioni dei più impegnati gruppi di ricerca italiani ed europei, fra le quali ha fatto spicco il gruppo di Parigi (tra cui Di Giugno - *IRCAM*) con una scheda popolatissima di LSI, del costo non indifferente di circa 40 milioni di lire, direttamente applicabile ai sistemi PDP della Digital.
Chi ha potuto partecipare al convegno - del quale esiste una completa relazione (si veda la bibliografia finale) - ha dunque assistito alla passerella più aggiornata delle tecniche usate oggi per "produrre" la Musica Elettronica.
È questo un consistente movimento di ricerca che effettivamente esiste, ed è quindi giusto offrire ai lettori di BIT la possibilità di accostarsi a questo tipo di attività.
Si tratta dunque, a questo punto, di focalizzare il problema che sarà risolto su queste pagine.
Innanzitutto la scelta di base: e cioè la descrizione di un sistema di controllo, tramite microcomputer, che sia immediatamente operativo e che richieda ben poco, oltre ad una normale struttura di sintesi analogica, per funzionare in modo completo.
Dunque non aspettatevi, da queste pagine, la minuta descrizione delle interfacce per PDP 11 o per IBM 370 usate da Di Giugno o da Grossi; ne sperate di ritrovare la completa teoria sulla autogenerazione di figure musicali complesse su basi di calcoli statistici: complete relazioni su queste ricerche sono infatti ritrovabili su appositi testi.
Su BIT, invece, troverete, oltre ad una accurata "fase di rodaggio" sul SYM-1 (il computer che useremo), tutti i particolari riguardanti la messa a punto di un controller che, applicato al sintetizzatore, renda possibile la polifonia più completa, essendo tutti i parametri sotto controllo del software.
Avendo scelto di sviluppare completamente questo argomento, i risultati attesi sono:
— la possibilità di "fraternizzare" col SYM-1 ed i suoi board di espansione grazie ad una applicazione di estremo interesse;
— la possibilità, per chi in effetti è veramente interessato alla Musica Elettronica oltre che al computer, di realizzare fino in fondo e con una spesa relativamente bassa il controller digitale;
— la possibilità, che spero vivamente sia accolta con partecipazione, dai lettori di BIT, di apportare ciascuno la propria esperienza in merito, in quanto a mio avviso l'argomento - per come è impostato - permette una tale partecipazione.
E, forse, dei tre risultati, il terzo è quello che propone una verifica più interessante.
Permettetemi, ora, di chiarire la maniera in cui voglio affrontare l'argomento.
Innanzitutto farò in modo che ogni lettore si accosti al SYM-1, entrando in confidenza con il board stesso e tutte le possibili espansioni.
Questo permetterà di affrontare la risoluzione del problema di controllo del sintetizzatore con basi più concrete.
Ove possibile, cercherò - per coloro i quali posseggono già un computer diverso dal SYM-1 di analizzare le varie routines e

PERCHÈ ANCHE IL PIÙ ESIGENTE NON HA DUBBI PER SCEGLIERE GENERAL PROCESSOR?

Perché la GP ha più esperienza. La GP è la **prima** azienda italiana ad aver prodotto microcomputers e personal computers; la prima in ordine cronologico e la prima per produttività. È anche la prima per la sua rapida espansione.

Perché la diffusione dei prodotti GP è conferma di qualità. I sistemi GP entrano anche "negli ambienti che contano". Esperti tecnici, istituti universitari, industrie, enti di ricerca (come il Consiglio Nazionale delle Ricerche) si affidano ogni giorno al nome GP

Perché i prodotti GP sono i più prestigiosi. Ogni progetto è fatto con in mente l'utente finale, i suoi problemi, le sue esigenze. Nessun dettaglio è trascurato e la scelta dei componenti è fatta in base a criteri estremamente rigorosi.

Perché la gamma dei prodotti GP è estremamente vasta: il nuovo Modello T è completamente espandibile in senso verticale:

★ Il T/05 con registratore audio, per l'hobby o per il calcolo scientifico

★ Il T/08 dotato di minifloppy disk per la più vasta gamma di problemi applicativi

★ Il T/10 destinato alla gestione di aziende di medie dimensioni con una estesa memoria a dischetti IBM compatibili

★ Il T/20 con un grande disco da 14 a 24 Mega bytes che vi aspettereste di trovare solo su un sistema di costo molto maggiore.

Perché i «personal» della GP dispongono di una delle più vaste biblioteche software del mondo: Il nuovo Modello T è compatibile col famosissimo CP/M (*), il più diffuso sistema operativo a dischi oggi esistente. Sotto il CP/M (*) sono disponibili tutti i più conosciuti linguaggi di programmazione; quindi non più soltanto il BASIC, ma anche FORTRAN, COBOL, APL, PASCAL, BASEX, ASSEMBLER ecc. ecc. Il servizio software della GP è poi a vostra disposizione per personalizzare secondo le vostre necessità i numerosi programmi applicativi già realizzati o per studiarne dei nuovi. Problemi già risolti includono la contabilità generale, la gestione del magazzino, la contabilità semplificata, la prenotazione elettronica degli appuntamenti, il listino prezzi on line...

Perché l'assistenza di una ditta che opera in Italia è per forza la migliore. Una garanzia che solo una ditta italiana può offrire: la certezza di una buona e completa assistenza.

Qualunque sia il problema la risposta è una sola: General Processor. La General Processor è vicina; telefona (al mattino) allo 0.55 - 21.91.43.

(*) trade mark of Digital Research, USA

SISTEMI DI ELABORAZIONE - MICROPROCESSORI
VIA PANCIATICHI, 40
VIA PIAN DEI CARPINI, 1
TEL. (055) 435527 - 50100 FIRENZE

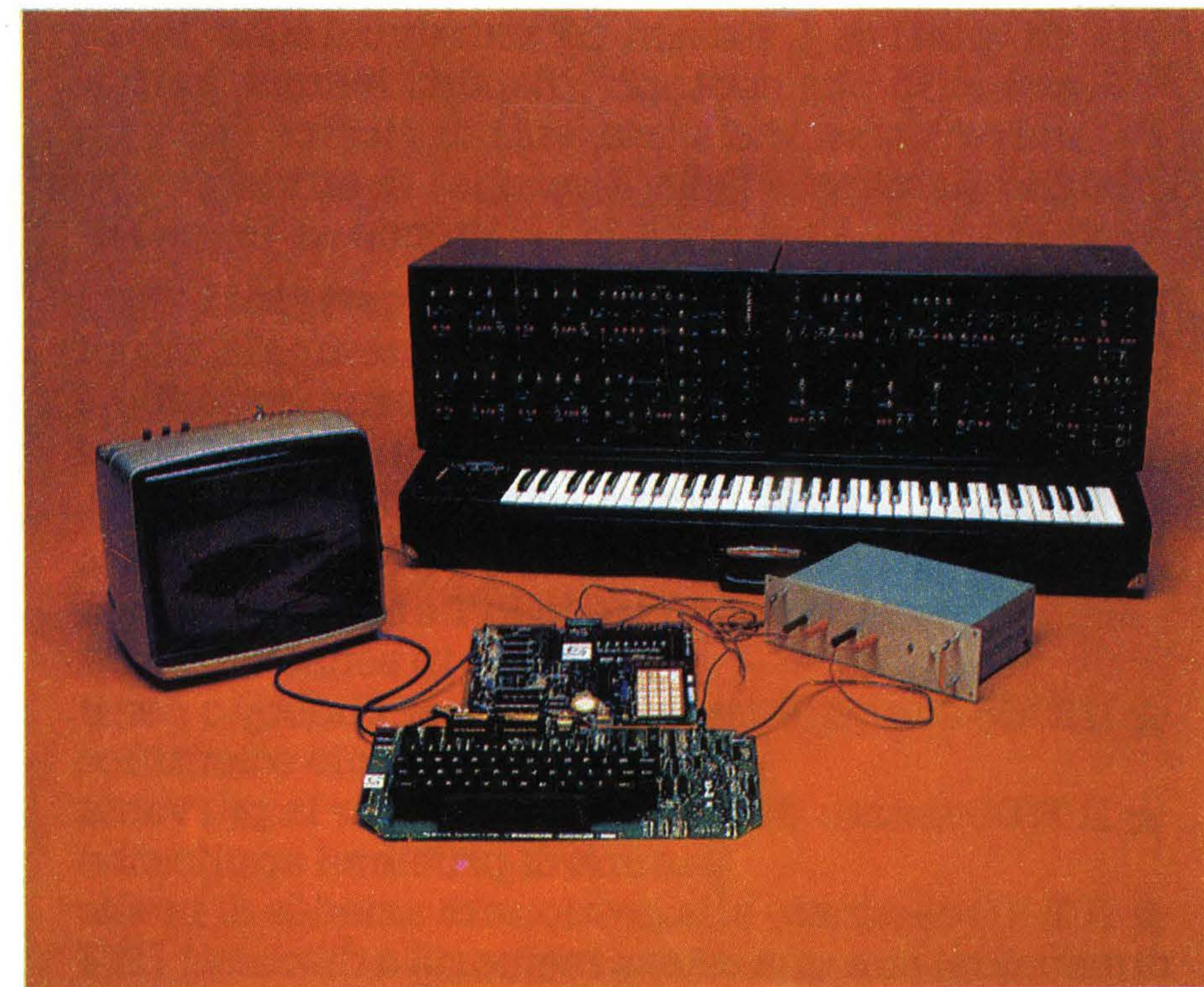
GP general processor

applicazioni mettendo in evidenza i punti comuni fra i vari board (pongo come pregiudiziale, perlomeno, che il computer usato sia basato su una CPU 6502 & Integrati della famiglia).
Seguirà quindi l'analisi della struttura di controllo, con la presentazione del convertitore Digitale-Analogico, dei circuiti di Multiplex/Sample-Hold, e, infine, del software che gestisce il controller stesso.
A questo punto spero di poter contare anche sulle vostre esperienze e quindi di poter proseguire, sulle pagine di BIT, al vostro fianco.

Alcune nozioni di Computer Music

Mi permetto di precisare alcuni punti che possono essere oscuri. Innanzitutto alcuni concetti elementari per chi si accosta per la prima volta all'affascinante argomento della Computer Music.
Ebbene, per eseguire della musica, occorre uno strumento.
La base fondamentale dell'esecuzione è l'emissione del segnale acustico sotto forma di onde sonore. È, questo, per esempio il suono del violino quando le sue corde siano eccitate dall'archetto.
Analogamente, con l'identico principio ma con metodi diversi, possiamo generare suoni con apparecchiature elettroniche amplificate è il caso di organi, tastiere d'archi, e così via. Tutto questo NON è Musica Elettronica, ovvero non lo è totalmente in quanto, per poter definire un suono *prodotto da un apparecchio di sintesi elettronica*, dobbiamo accedere a macchine che mettano a disposizione dell'utente il diretto controllo di TUTTI i parametri del suono, vale a dire il contenuto armonico (*timbro*), la frequenza caratteristica (*tono*) e la maniera con cui il suono nasce, si sviluppa e quindi muore (*dinamica*).
Siamo così giunti al punto di dover operare una distinzione.
La prima strada vede l'uso di un elaboratore su cui impostare, tramite software (molto) e hardware esterno (pochissimo, in genere solo convertitori Digitali-Analogici) la nostra figura musicale.
Una volta inserito il programma l'elaboratore lo compila e quindi può avvenire l'esecuzione.
È evidente come una produzione di Musica Computerizzata sotto totale controllo del software permette una enorme versatilità nell'esercizio, permettendo all'utente di cambiare - con piccole modifiche di software - parametri quali ampiezze, frequenze, tempi di battuta, e così via. L'adozione di una tale tecnica di sintesi, inoltre, permette lo studio e la messa a punto di appositi "editors" musicali, ovvero linguaggi specifici per la traduzione delle partiture o per l'elaborazione autonoma - il computer stesso che compone - della Musica. A titolo informativo, posso rimandarvi in proposito ai lavori di Pietro Grossi.
Il contro di una tale impostazione si riassume nei seguenti punti:

- necessità di un elaboratore veloce e piuttosto complesso; quindi a tale ricerca possono accostarsi solo coloro che stabilmente partecipano ad un gruppo di ricerca che possa usare un grosso elaboratore;
- svantaggi nelle applicazioni in tempo reale; se è infatti facile (relativamente!) eseguire partiture o produrre figure musicali in genere preprogrammate, la Musica Elettronica in tempo reale, qualora sia generata esclusivamente in modo digitale, è molto difficile da realizzare.

Soltanto da poco, infatti, i ricercatori dell'IRCAM hanno messo a punto una interfaccia di sintesi completamente digitale, ma il costo elevatissimo - per ora, forse - riduce sensibilmente la fascia degli utenti possibili.
Non trascuriamo, infine, il fatto che il discorso di affidarsi completamente al computer per la generazione di Musica Elettronica decade completamente quando si usa un microcomputer anche se diligente, nè il nostro SYM nè alcun altro microcomputer è assolutamente in grado di gestire programmi di generazione di M.E. (Musica elettronica) esclusivamente prodotta digitalmente (per campionamento, all'uscita di un convertitore Digitale - Analogico applicato ad un Port di uscita); se non in forme molto ridotte (massimo 4 voci e $F_{max} = 3$ kHz) ed assolutamente NON in tempo reale.
Dunque escludiamo fin d'ora la possibilità di sederci di fronte al nostro SYM, introdurre un programma più o meno gigantesco, collegare l'amplificatore e ... suonare.
La seconda strada - quella da noi seguita - vede invece l'adozione del computer come versatile e potente controllore di una struttura di sintesi fondata su normali moduli analogici.

Vista del SYM e del sintetizzatore audio che sarà dal SYM stesso controllato, per la realizzazione della polifonia.

Ecco, dunque, che entra in scena il SINTETIZZATORE. Quest'ultimo è una macchina prettamente analogica, ovvero formata da moduli analogici che possono essere collegati fra di loro in varie maniere, tramite commutatori o cavetti, in modo che l'utente crei con le sue mani il suono voluto.
Dei moduli usati normalmente, citerò gli *oscillatori*, che generano le forme d'onda che poi i *filtri* dovranno elaborare, mentre degli appositi *amplificatori*, il cui guadagno è controllabile esternamente, provvedono a forgiare la dinamica del suono generato.

In genere, inoltre, in tali strutture analogiche è sempre presente una tastiera tipo pianoforte, che permette all'utente - quasi sempre musicista - di produrre suoni armonicamente correlati fra di loro e legati ovviamente alla specifica posizione dei tasti premuti sulla tastiera. Detto questo, sembrerebbe che una tale struttura possa be-

nissimo agire anche in completa autonomia.
E così è, infatti: l'enorme successo del sintetizzatore nei gruppi musicali dimostra quale sia il peso del suono "elettronico" nella musica moderna.
C'è da dire però, che il sintetizzatore analogico, per la natura sua e dei circuiti con cui è costituito, è uno strumento MONOFONICO. Ovvero l'utente può suonare solo una nota alla volta.
Questa è una fortissima limitazione, soprattutto per il musicista: da qui lo sviluppo di moltissime tecniche volte a permettere una completa POLIFONIA dello strumento, pur lasciando al sintetizzatore la sua eccezionale pecularietà di macchina POLITONICA.
Ovvero un organo elettronico è uno strumento *polifonico* perchè vi si possono suonare - per esempio - degli accordi: ma non è *politonico* in quanto l'utente non può in alcun modo attribuire solo a certi tasti un timbro e ad altri un altro.
Invece un sintetizzatore è e *deve* restare anche politonico.
Avere la polifonia su una tale macchina, dunque, significa disporre di un sistema dalla versatilità enorme e capace di produrre le varietà più incredibili di suoni.
Orbene - e qui ritorniamo al nostro programma, - ecco che uno dei metodi per realizzare la polifonia su di una struttura monofonica è proprio l'adozione del computer come controllore generale del sistema.
Inoltre ricordiamoci che le modalità del controllo sono sotto la giurisdizione del nostro software, per cui è solo la nostra (e vostra) capacità di realizzare programmi di controllo flessibili che limita la versatilità della macchina una volta che sia collegata al computer.

Per ... gli interessati, tutto ciò che appare in questa fotografia è usato e sarà usato dall'Autore per i suoi diabolici esperimenti.

Restano da dire ancora alcune cose, prima di passare alla parte seguente ove inizieremo a scoprire il SYM.

Il "nostro" programma di lavoro

La prima cosa è un avvertimento, più che altro rivolto a chi crede di vedere un programma statico ed impostato sul fate questo e questo. In molte occasioni lancerò delle proposte, quasi degli "esercizi da fare a casa".
Spero in questo di trovare un apprezzamento e - mi ricollego a quanto detto precedentemente a proposito delle vostre esperienze - mi auguro che il lettore di BIT capisca che con un poco di buona volontà è possibile contribuire alla sua rivista.
La seconda cosa riguarda ancora il programma, ed è un consiglio su come affrontarlo.
I problemi legati alla Computer Music, infatti, sono tanti e tali che, se dovessi usare solo BIT per descrivere tutto il sistema ed i suoi accessori, non basterebbero tre o quattro numeri.
Da qui il consiglio di seguire il mio programma "gemello", che dal numero di dicembre/gennaio '80 appare sulle pagine di selezione-Radio TV.
In quella serie di articoli troverete descritti in maniera chiara e spero dettagliata tutti i componenti della struttura analogica più propriamente detta, vale a dire oscillatori, filtri, etc.
Infatti per come abiamo impostato il nostro programma (ovvero l'applicazione del computer ad una struttura analogica standard), dovrete per forza avere un sintetizzatorte analogico.
Dunque Selezione è il "manuale" di applicazione ove trovare le proposte e gli esempi onde costruire il vostro sintetizzatore.
Altrimenti potrete sempre rivolgervi a strutture commerciali, quali Roland System o Moog, ma tenete conto della ... spesa globale.
Riassumendo: BIT per il software e Selezione per l'hardware, soprattutto analogico.
In più BIT evidenzierà quella parte della Computer Music che è più strettamente correlata con il solo computer. Ciò in quanto molte - seppure semplici - applicazioni possono essere realizzate con l'uso del solo SYM (sfruttando i timers interni, o con l'aggiunta al massimo di una schedina di conversione A-D).

Riassumendo, vedremo dunque la Computer Music prodotta dal connubio fra Computer e Sintetizzatore Analogico; vedremo applicazioni del computer nella generazione di suoni *non colorati* in monofonia ed in polifonia; vedremo anche alcuni aspetti del possibile controllo di altre tastiere (ad esempio, d'archi o organi o piano elettrico) col computer, usando vari tipi di interfaccia.
Concludo qui questa lunga chiacchierata di introduzione, raccomandandovi la piena libertà di esprimere i vostri pareri e/o consigli scrivendomi presso la Redazione di BIT.
A questo punto lasciamo da parte - per ora - la Musica Elettronica ed incominciamo a conoscere il computer che useremo: il SYM e le sue periferiche.

Figura 1 - Architettura interna dei microprocessori della serie 6500.

Il 6502

Prima di parlare del SYM, è bene avere un'idea delle caratteristiche della CPU che esso utilizza, il 6502: la velocità (ciclo di 1 μsec), i modi di indirizzamento (ben 13), ed il set di istruzioni (56) che non sono una valanga ma permettono un flessibilità di esercizio incredibile (ed un accostamento al chip molto veloce ed indolore) rendono questo microprocessore estremamente interessante (*v. Figura 1*).
Inoltre la struttura interna, e il suo interfacciamento con il mondo esterno, sono altri punti di forza del 6502.
Innanzitutto è utile far notare la facilità del 6502 nel gestire IRQ, NMI, RESET (restart): i pins appositamente dedicati, al mutare della condizione applicata, istruiscono il microprocessore, in ogni caso, a salvare il registro Program Counter, il Registro di Stato inpilandoli nello stack), e quindi caricare nel Program Counter gli indirizzi dei *vettori* di IRQ o MNI o RESET: tali vettori implicano che venga a sua volta puntata una cella di RAM o ROM ove effettivamente risiede l'indirizzo di partenza della sequenza di restart, della routine di gestione dell'NMI o dell'IRQ.
Detto in parole più povere, facciamo l'esempio dell'utente che usi il 6502 in un computer "dedicato", e che quindi debba assicurarsi che, all'atto della accensione, la sua scheda effettivamente inizi il lavoro per cui è stata costruita.
Ebbene, nulla di più facile: infatti sappiamo (penso di parlare a lettori che già conoscono almeno un poco il 6502: se no, niente di meglio che dare un occhiata al manuale hardware relativo (vedasi bibbliografia) che, all'accensione, la logica hardware esterna che dovremo aver cablato (il solito monostabile) manda la linea RESET a zero per qualche istante.
Percepito questo, la CPU azzera tutti i registri, si assicura che non venga abilitata alcuna scrittura verso l'esterno, quindi carica nel PC (registro Program Counter) FFFC (hex), che deve ovviamente essere una casella di ROM (in ogni caso non volatile). Qui si dovrà trovare la parte *bassa* dell'indirizzo che, per esempio, chiameremo "RESET LO" (fantasia, no?). La parte *alta*, ovvero "RESET HI", sarà contenuta in FFFD.
Dunque l'indirizzo "RESET" sarà quello di una cella - nel nostro caso ROM - dalla quale parte effettivamente la sequenza di Reset (o Restart).
Una tipica sequenza inizierà, per esempio, con la sistemazione corretta dello stack pointer, seguita da un corretto setup delle periferiche, a sua volta seguita, infine, dalla vera e propria routine di lavoro del computer.

Senza scendere nei particolari, dirò che la gestione degli interrupt avviene nella medesima maniera, riservando gli indirizzi FFFE, FFFF e FFFA, FFFB per IRQ e NMI.
IRQ e NMI sono linee che, se progettate ad OR cablato come è solito possono sfruttare perfettamente il modo di vettorizzare gli interrupt del 6502, in

quanto, per la gestione polling degli interrupt, basterà che il programmatore realizzi una opportuna routine *puntata* dal vettore di interrupt abilitato.
Un'altra particolarità della struttura del 6502 è il modo, veloce e semplice, di gestire Read e Write: il clock a 2 fasi (Φ1 e Φ2) e la sola linea R/$\overline{W}$ permettono questo.
In fase 1, infatti, sul bus degli indirizzi appare l'indirizzo della cella chiamata: è dunque in questa fase che possono essere generati tutti i segnali di abilitazione. In fase 1, inoltre, viene generato il segnale di R/$\overline{W}$: un uno logico indica alla memoria (in generale a ciò con cui si comunica) che l'operazione riguarda una *lettura*; uno zero logico abilita invece una *scrittura*. Una volta che tali segnali si sono stabilizzati, finalmente, in fase 2, appare il dato sul bus-dati. È dunque in fase 2 che fra il 6502 e tutte le periferiche avvengono gli scambi di dati ed informazioni varie (*v. Figura 2*).
Riassumendo, su ogni interfaccia che comunichi con la CPU 6502 devono giungere: linee dei dati, linee degli indirizzi, R/$\overline{W}$, fase 2 del clock, linea di selezione (chip select).
Per concludere il nostro discorso sul 6502, accenerò alle periferiche che compongono la famiglia 6500.
Infatti un altro dei notevoli punti a favore della scelta del 6502 risiede proprio nella presenza di una enorme famiglia di componenti di interfaccia, che, come il 6502 stesso, godono di invidiabili caratteristiche di velocità e versatilità.
Un breve elenco (per maggiori dettagli rimandiamo alle pubblicazioni Synertek, Mostechnology e Rockwell) mostra che, oltre ai vari tipi di memorie veloci, Eprom ad una sola alimentazione (da 1, 2, 4 k esistono:

— SY/MPS 6522: è la VIA (*V*ersatile *I*nterface *A*daptor), integrato a 40 pins con le funzioni di doppio port di I/O, shift register e doppio timer/counter (*v. Figura 3*). Ogni funzione è sotto controllo del software. Di caratteristiche simili (ma diferenti nelle strutture interne) sono quindi le 6520/6530/6532: ovvero I/O & Timer, oppure I/O & RAM e/o ROM, etc.
— SY/6545: è il CRT Controller, integrato a 40 pins per il controllo più completo dell'interfacciamento con video.
— SY/6551: ACIA (*A*synchronous *C*ommunications *I*nterface *A*daptor), integrato a 28 pins che gestisce autonomamente le comunicazioni seriali in full duplex con Baud Rate programmabili da 50 a 19.200 Baud.

E, con questo, fatta una breve ma spero completa panoramica sul 6502, si conclude la prima parte del mio programma, che proseguirà con la descrizione del SYM-1 e delle sue periferiche.
Diamo ora la bibliografia essenziale per i problemi che saranno discussi in seguito. ■

Figura 2 - Diagrammi temporali per i microprocessori della serie 6500.

Figura 3 - Schema a blocchi dell'architettura interna della VIA 6522.

Bibliografia

1) *Microcomputer Components Databook* & Datasheets on: 2114, 6502, 6522, 6532, 6545, 6551 (Synertek, Inc; Rockwell, Mostechnology)
2) *Computer music*, "BYTE Publications, Inc."
3) Articoli sulla Polifonia, in "Polyphony Review" (1977/78/79/80).
4) *System Databook for VEB Card* (E - mu Systems, Inc.).
5) *5600/3800/4600 System DataBook (ETI - International Today-Press).*
6) *Bozzola Paolo, Progettazione, costruzione di un sistema di un sistema completo di moduli per la sintesi audio; controllo a microcomputer* (dispense).
7) *Musica ed elaboratore elettronico* (Relazione sul convegno omonimo, Aprile 1980. FAST, Milano).

Il calcolatore per i giovani: i numeri a virgola mobile

di **F. Waldner**
Università di Bari, Istituto di Fisica

Parte III

Fino a questo punto abbiamo preso in considerazione solo numeri *interi* (integers), detti anche *a virgola fissa* (fixed point). Evidentemente parecchi di voi avranno già in mente una obiezione: come si fa a fare operazioni tipo 1,2 + 7,5? Cioè, come si può operare su dei numeri che *non* sono interi, senza dover ricorrere a salti mortali tipo quelli di moltiplicare per 10 o 100 gli operandi e di dividere poi il risultato per lo stesso numero?

Queste operazioni possono in realtà venire eseguite dal calcolatore senza alcuna difficoltà, anche se è da notare che sono operazioni che richiedono *più tempo* delle corrispondenti operazioni sui numeri interi. Questo è un punto che dovete tener ben presente nei vostri programmi.
Prendendo come pietra di paragone un minicomputer di alta classe (il PDP-11, della Digital Equipment Corporation), vi posso dire che una somma di due numeri interi richiede 5,17 μs = 5170 ns, cioè un tempo ben 17 volte più lungo. Le cose diventano meno drammatiche per operazioni più complesse. Per esempio la moltiplicazione fra interi richiede 3,3 μs, mentre fra non interi richiede 7,17 μs; per la divisione (l'operazione più complicata) le cose vanno ancora meglio: fra interi sono necessari da 6,9 a 7,5 μs, e fra non interi da 8,17 μs a 11,87 μs.
In ogni caso questi numeri insegnano una cosa: è bene usare nei programmi quanto più è possibile i numeri interi, ed usare i *floating point* solo se è effettivamente richiesto dal calcolo. Fino a dove è possibile il numero intero va sfruttato, in quanto permette la massima velocità di calcolo.
Vediamo quindi come vengono rappresentati i numeri *a virgola mobile*, detti anche *reali* (reals) o *floating point* (FP d'ora in avanti).

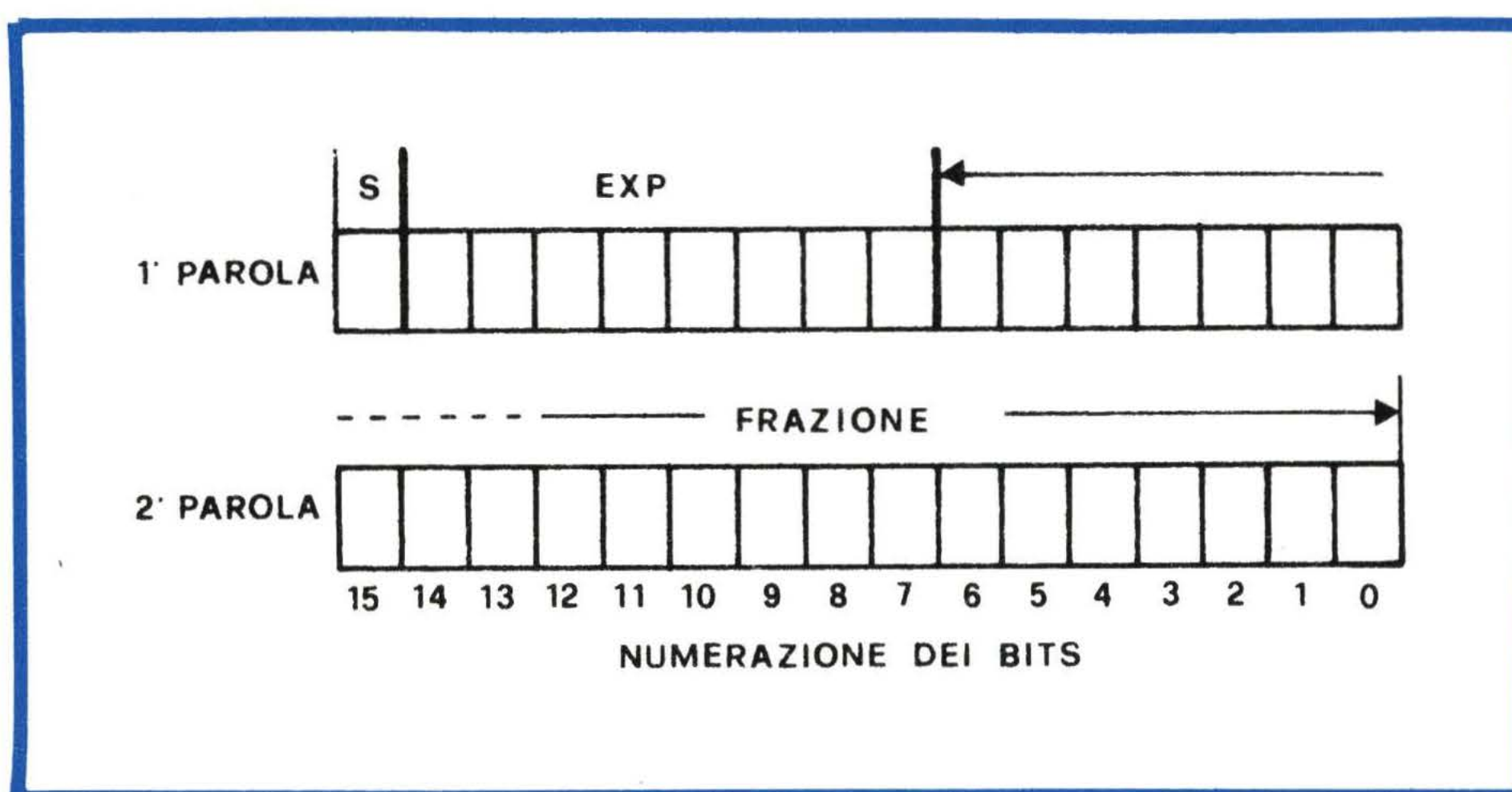
S EXP
1ª PAROLA
FRAZIONE
2ª PAROLA
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
NUMERAZIONE DEI BITS

Figura 1 - Rappresentazione di un numero reale nella memoria di un calcolatore.

La rappresentazione dei numeri a virgola mobile

Anzitutto va detto che per un FP non bastano più *due* bytes, ma ne occorrono *quattro* (cioè due parole intere da 16 bits), e questo se si vuole avere la possibilità di ottenere dei risultati con 8 cifre decimali. E non crediate che questa sia una precisione eccessiva ed inutile non lo è per le seguenti ragioni:

— anzitutto perchè nel corso dei calcoli succede in pratica che le ultime due cifre sono usate praticamente solo per arrotondamenti (e quindi in realtà noi possiamo contare su 6 cifre decimali esatte);
— in secondo luogo perchè 6 cifre decimali esatte (cioè la precisione di una parte su un milione) non sono poi eccessive: capita anche troppo spesso in calcoli appena appena un pò complessi che la precisione ora descritta sia completamente insufficiente, e che sia necessario "aumentare l'informazione".

Si arriva così a dei floating point che vengono descritti con 8 bytes, cioè con 4 parole a 16 bits: questi numeri vengono anche detti numeri reali *in doppia precisione* (double precision, o DP). L'uso di rappresentazione di questo genere è evidentemente da "centellinare", dovendo esso essere limitato ai casi in cui sia veramente indispensabile. E ciò non tanto - badate bene - per il tempo che è richiesto alle singole operazioni: sorprendentemente infatti le operazioni in DP sono solo dal 30 al 50% più lunghe delle corrispondenti in FP.
Il problema in realtà è la tremenda occupazione di memoria del calcolatore che un DP comporta; nella memoria richiesta da un solo DP troverebbero posto infatti ben 4 numeri interi.
D'altro canto è bene che teniate presente che i numeri DP non sono relegati in laboratori superspecializzati nei quali si fanno chissà quali *calcoli astrusi*. In realtà può capitare anche a voi di doverli usare (e nei prossimi articoli vedremo anche qualche esempio in merito) non appena abbiate a che fare con *differenze di numeri quasi uguali*. E questo può capitare più spesso di quanto non crediate: basta un modesto sistema di tre equaazioni in tre incognite per potervi mettere di fronte alla necessità di usare numeri in doppia precisione, a meno che non vogliate ottenere dei risultati a dir poco discutibili.
Un numero reale occupa quindi 4 bytes, che tipicamente sono suddivisi come rappresentato in *Figura 1*.
Come vedete, i 4 bytes sono in effetti trattati come una "striscia continua" in cui il calcolatore attribuisce particolari significati ai bits a seconda della loro posizione.
Vediamo così che, da sinistra a destra, il primo bit è riservato al segno del numero reale in esame. I successivi 8 bits sono riservati all'*esponente*, e tutti gli altri alla *frazione*, o *mantissa* se preferite. Questi due termini richiedono un attimo di spiegazione.
Capita spesso nel calcolo scientifico e tecnico di dover lavorare con numeri straordinariamente grandi o straordinariamente piccoli, dal momento che non si possono cambiare ogni momento le unità di misura. Essenzialmente per questa ragione è stata introdotta la cosidetta *notazione scientifica* (cui abbiamo già fatto cenno nell'articolo scorso) che permette di trattare numeri grandi o piccoli senza dover incorrere negli inconvenienti della notazione decimale solita: è indubbiamente più comodo scrivere $1{,}06 \cdot 10^{-13}$ invece di 0,000000000000106.

Il calcolatore per i giovani

È proprio la notazione scientifica che viene usata per rappresentare i FP nel calcolatore, ma prima di dirvi come si ottengono le varie parti di cui è composto il FP, sarà bene fare un esempio col sistema decimale, col quale tutti abbiamo una certa familiarità.
Detto incidentalmente: non crediate che il sistema decimale sia poi un modo facile di rappresentare i numeri! Pensate alla fatica che avete fatto voi alle elementari (e la vostra maestra ed i vostri genitori assieme a voi...) e vi renderete conto che il sistema decimale è un tipo di notazione estremamente raffinato e sofisticato, e che si giustifica molto bene il fatto che all'umanità ci sia voluto molto tempo per scoprirlo e metterlo a punto.
Supponiamo dunque di voler rappresentare il numero

— 5.748,32

Anziutto questo potrà venir scritto

$-0,574832 \cdot 10^3$

stabilendo una volta per tutte la convenzione di aggiustare il fattore con la potenza di 10 (detto anche "ordine di grandezza") in modo tale che il primo fattore (mantissa) sia *sempre* compreso fra 0 e 1. In tale caso diremo che la mantissa è *normalizzata*. *Se* siamo d'accordo sul fatto di adoperare *sempre* questa convenzione, allora possiamo vedere che nel numero così scritto alcune parti sono inessenziali, in quanto sono sempre presenti in ogni numero, e ripeterle ogni volta è un inutile spreco di carta (e di informazione).
Infatti ecco alcuni numeri scritti nella notazione or ora descritta:

$0,756845 \cdot 10^4$
$-0,579456 \cdot 10^3$
$0,395845 \cdot 10^5$

Potete vedere che *tutti* i numeri cominciano con 0, e *tutti* i numeri hanno l'esponente della potenza di 10.
Possiamo quindi compattare la scritture convenendo di scrivere *solo* l'esponente e *solo* la frazione, oltre al segno di quest'ultima. In questa nuova notazione i numeri precedenti diverrebbero come in *tabella I*.
Naturalmente non sarò io a propugnare che d'ora in avanti si insegni nelle scuole a scrivere i numeri in questo modo, però il calcolatore - ricordatelo! - non ha problemi di apprendimento di notazioni astruse, mentre ha problemi di spazio e di efficienza di gestione del tempo. Quindi una notazione come quella descritta che - notate! - riduce con opportune convenzioni dei numeri decimali *a dei numeri interi*, per un calcolatore può andare benissimo.

Resta però ancora un punto da chiarire: supponiamo che l'esponente sia *negativo*, il che poi corrisponde ad avere un numero *minore di 1*. Per esempio:

$0,678945 \cdot 10^{-3}$

Come ce la caviamo in questo caso, ovvero più in concreto, dove possiamo sistemare l'informazione del *segno dell'esponente*? Potremmo - è vero - ricorrere alla notazione del "complemento a due", secondo l'aurea ricetta del Tesoriere (di cui al numero 7 di BIT).
In realtà si preferisce una notazione diversa, che possiamo descrivere nel nostro caso in questo modo: supponiamo che scrivere per l'esponente il numero 5 voglia dire *esponente 0*, scrivere 6 voglia dire *esponente 1*, scrivere 4 voglia dire *esponente -1*. In pratica supponiamo di usare una tabellina del genere della *tabella II*.
Una notazione di questo genere si dice *in eccesso di 5* e permette di scrivere esponenti da -5 a +4 con un singolo numero intero.
Questo stesso tipo di filosofia viene mantenuto pari pari nella notazione FP. Il primo bit indica il segno della mantissa. I successivi 8 bits indicano l'esponente, in notazione *in eccesso di 128* (cioè in eccesso di 200^8*) e permettono di rappresentare degli esponenti (in base 2) che vanno da un minimo di -128 a un massimo di* +127. Col che è evidente che si possono rappresentare numeri fantasticamente piccoli ed immensamente grandi: il fatto che l'esponente sia 127 vuol dire che il più grande numero che si può rappresentare ha come ordine di grandezza $2^{127} = 1,7 \cdot 10^{38}$. Quindi si possono rappresentare in questo modo dei numeri che vanno da 10^{-38} a 10^{+38}.

Un'ultima raffinatezza consiste nel fatto che - come nel caso della rappresentazione decimale che abbiamo analizzato prima - la mantissa è *normalizzata*. Ora questo non vuol dire molto nel sistema decimale: in pratica vuol dire solamente che la prima cifra della mantissa deve sempre essere una cifra che va dall'uno al nove. In altri termini *non può essere zero*. Nel sistema binario però la faccenda è completamente diversa: dire che un numero non può essere zero, vuol dire che *deve* essere a 1. Conclusione: il primo bit della mantissa quando questa viene normalizzata *deve essere uguale a 1*. Ed allora è inutile scriverlo: *tanto vale sottointenderlo*. Infatti si fa così per guadagnare ancora posta per l'informazione essenziale eliminando fino a dove possibile quella inutile.

Un esempio

State tranquilli: stavolta non ci saranno problemi sulla notazione FP. In realtà è importante sapere come funziona tutto il marchingegno, ma raramente è necessario metterci le mani direttamente, per cui è inutile perderci troppo tempo a fondere le bronzine. Tuttavia vedremo un esempio di come si può risalire al numero in notazione decimale partendo da ciò che sta dentro il calcolatore.
Supponiamo quindi di avere un FP rappresentato da due parole in questo modo;

1 10101101 0101110
1110100000001111

Le spaziature sono state messe per mettere in evidenza le diverse parti in cui il numero deve venir scomposto.
Anzitutto il bit del segno è 1 e quindi il numero è negativo. Poi l'esponente (in base 2!) si trova negli 8 bits successivi, e vale

$10\ 101\ 101 = (255)_8$

come non vi sarà difficile appurare, ora che siete diventati esperti di sistemi binari, ottali ed esadecimali.
Ricorderemo quindi che l'esponente è scritto *in eccesso di* $(200)_8$, e ne dedurremo che esso quindi vale $(55)_8 = (45)_{10}$. Quindi il numero ottenuto dalla mantissa dovrà venir moltiplicato per 2^{45}.
Vediamo ora la mantissa. Questa è una lunga stringa di bits, ai quali va aggiunto il primo bit della catena, *che è sempre 1 e che è sottointeso*. Ecco qui la stringa di bits, incluso il primo:

101 011 101 110 100 000
001 111

che si ottiene partendo dal bit successivo all'esponente e scrivendo un bit dopo l'altro: finita la prima parola, si attacca coi bits della seconda.

Io ho aggiunto il primo bit (*sempre* uguale a 1!) ed ho diviso i bits in gruppi di tre, in modo che ora risulta semplice rappresentare la mantissa in ottale; eccola:

$(5\,3\,5\,6\,4\,0\,1\,7)_8$

Che vuol dire questo numero? In base 8 (molto più comoda

Con riferimento alla notazione scientifica, in cui la mantissa è normalizzata, i numeri

$0{,}756845 \cdot 10^4$ $-0{,}579456 \cdot 10^3$ $0{,}395845 \cdot 10^5$

si differiscono nei seguenti elementi:

segno	esponente	mantissa normalizzata
+	4	756845
-	3	579456
+	5	395845

Ponendo la convenzione che la prima cifra rappresenta *l'esponente* di 10 e le altre stanno a significare la *mantissa normalizzata*, potremmo anche scrivere tutto quanto di fila, ottenendo così:

+4756845 -3579456 +5395845

Tabella I

Si scrive	quando l'esponente è
0	-5
1	-4
2	-3
3	-2
4	-1
5	0
6	+1
7	+2
8	+3
9	+4

Tabella II - Scrittura di un esponente compreso tra 0 e 9 in "eccesso di 5".

per noi che non la base 2, che sarà meglio lasciare al calcolatore) vuol dire proprio questo:

0,53564017

Ma attenzione: questo è un numero "con la virgola" in base 8, *non* in base 10! Più precisamente: così come quando scriviamo 0,375 vogliamo indicare $3 \cdot 10^{-1} + 7 \cdot 10^{-2} + 5 \cdot 10^{-3}$, nel caso della mantissa ottale scritta in precedenza avremmo:

$5 \cdot 8^{-1} + 3 \cdot 8^{-2} + 5 \cdot 8^{-3} + 6 \cdot 8^{-4} + 4 \cdot 8^{-5} + 1 \cdot 8^{-7} + 7 \cdot 8^{-8}$

Se a questo punto impugnate il vostro calcolatorino tascabile per convertire questa espressione in un numero decimale, dovreste ottenere 0,683228433 nel sistema decimale.
Il nostro numero, che *in base 8* si poteva scrivere

$$-0{,}53564017 \cdot 2^{55}$$

in base 10 diviene

$$-0{,}683228433 \cdot 2^{45} = -0{,}2403896328 \cdot 10^{14}$$

che è il risultato che cercavamo.
Ecco quindi come dovete fare se doveste avere necessità di convertire il numero così come è scritto all'interno di un calcolatore in termini umanamente comprensibili. Vedete che *non* è un procedimento semplice: in effetti il mio sincero augurio è che vi capiti di doverlo fare il più raramente possibile. Anche perchè se proprio fosse indispensabile farlo vorrebbe dire che siete decisamente nei guai col vostro programma ....

Le espressioni nel linguaggio BASIC

Variabili e costanti possono essere variamente combinate con le *operazioni algebriche*, in modo da costituire *espressioni* che sono nella loro forma esteriore molto simili a quelle cui siete stati abituati fin dalle scuole medie.
I simboli che si usano sono sempre i soliti:
- I segni + e - per indicare somme e sottrazioni.
- Il simbolo * (asterisco) per indicare la moltiplicazione.
- Il simbolo / per indicare l'operazione di divisione (attenzione: *non* si usano i "due punti"!).
- La "doppia stella" (**) per indicare l'operazione di elevamento a potenza. In altri termini, se dovete indicare l'operazione 4^3 dovete scrivere 4**3 affinchè il calcolatore vi capisca.
- Le parentesi, le quali sono solamente *tonde*, e che hanno esattamente il significato delle espressioni solite: cioè di dare la precedenza all'esecuzione delle operazioni indicate nelle parentesi più interne, e successivamente alle altre.

Come viene eseguita un'espressione dal calcolatore? Anzitutto in base a quanto detto nell'ultimo paragrafo partendo dalle parentesi più interne. Comunque, all'interno di una parentesi, esiste una precisa gerarchia; le operazioni vengono eseguite nel seguente ordine:
- operazioni di elevamento a potenza
- prodotti e divisioni
- somme e sottrazioni

L'ordine - quindi - è *esattamente* quello previsto dall'algebra. E questa - badate bene - non è una scelta del BASIC, ma una scelta comune a tutti i linguaggi ad alto livello. Se leggete (come mi auguro) gli articoli sul Pascal vi accorgerete che anche in questo linguaggio le convenzioni sono le stesse.
A proposito di Pascal: consiglio ai più volenterosi di voi di porre un pò di attenzione e di calma nel seguire sia questo ciclo di articoli che quello sul Pascal. Infatti imparare un linguaggio è in fondo come imparare una lingua, e imparare *due* lingue contemporaneamente può porre dei problemi .... Niente paura: sono problemi facilmente superabili con un pò di organizzazione, buona volontà e metodo. In ogni caso è bene che prendiate fin dall'inizio tutte quelle precauzioni atte ad evitarvi di "fare confusione". Cosa - detto fra noi - molto più facile di quanto non si creda.
Torniamo al nostro BASIC e vediamo una serie di esempi su come si possono scrivere alcune formule note (*spero*!) a tutti voi.

Esempi

Lunghezza della circonferenza

Sapete che la formula è:

$$L = 2 \pi R$$

Potrete tradurre questa formula in BASIC in uno dei seguenti modi:

C = 2 * 3.141593 * R

oppure

C = 6.283185 * R

Nel primo caso lascerete alla macchina la cura di effettuare per conto vostro il prodotto 2 π.

Il calcolatore per i giovani

Nel secondo caso gliela risparmierete. Mi chiederete quale dei due metodi sia preferibile. La risposta è: "dipende". Mi spiego meglio: se volete un programma che sia più facilmente comprensibile *a voi* quando lo rileggerete magari fra tre mesi, e quindi vi sarete abbondantemente dimenticati dei dettagli, e se non vi importa molto del fatto che la macchina faccia qualche operazione in più e quindi ci metta un pò più di tempo, allora usate pure il primo sistema.
Se però siete sicuri di voi stessi (e magari per non dimenticarvi di ciò che avete fatto usate la buona abitudine di inserire qua e là nel vostro programma qualche commento, per ricordare a voi e ai vostri colleghi ciò che avete in mente in questo istante) allora usate pure il secondo sistema: la macchina impiegherà meno tempo per giungere allo stesso risultato.

Area del cerchio

La formula che ci insegnano è:

$$A = \pi R^2$$

che possiamo tradurre in BASIC nel seguente modo:

S = 3.1415927 * R ** 2

Qui si impone una osservazione. Alcuni di voi potrebbero obiettare che è inutile usare una così accurata approssimazione di π quale quella qui descritta, quando nelle scuole si insegna ad usare 3,14 o al massimo 3,1416... Ebbene: è un errore. Tenete presente che la macchina impiega lo stesso tempo a moltiplicare per 3,14 che a moltiplicare per 3.1415927. Per voi, armati di carta e matita, la cosa è ben diversa, ma per il calcolatore no: il tempo impiegato *è lo stesso*, ed è quello impiegato nella moltiplicazione fra due numeri in virgola mobile. Ed allora è inutile perdere in precisione, senza guadagnarci da nessuna altra parte. Tanto vale usare per le costanti la massima precisione possibile, e lasciare che la macchina lavori: è stata fatta apposta per alleviare la fatica umana, o no?
Seconda nota: mi dimenticavo di dirvi che in BASIC, come in tutti gli altri linguaggi, potete inserire *spazi* (si chiamano *blanks*) dove volete, in modo da avere delle stampe più leggibili. Quando siete in presenza di un calcolatore però non dimenticate di dare un'occhiata se per caso non ci siano delle restrizioni. Potrebbe capitarvi di *non potere* inserire spazi in certi posti, magari all'interno di una costante: per esempio potrebbe essere proibito scrivere

3.13 15927

Ciò può dipendere dal particolare sistema col quale state lavorando, e quindi tenete presente queste possibili limitazioni.

Volume della sfera

La formula è:

$$V = 4/3\ \pi\ R^3$$

che potremo tradurre nel seguente modo:

V = 4/3 * 3.1415927 * R ** 3

oppure nella più compatta notazione

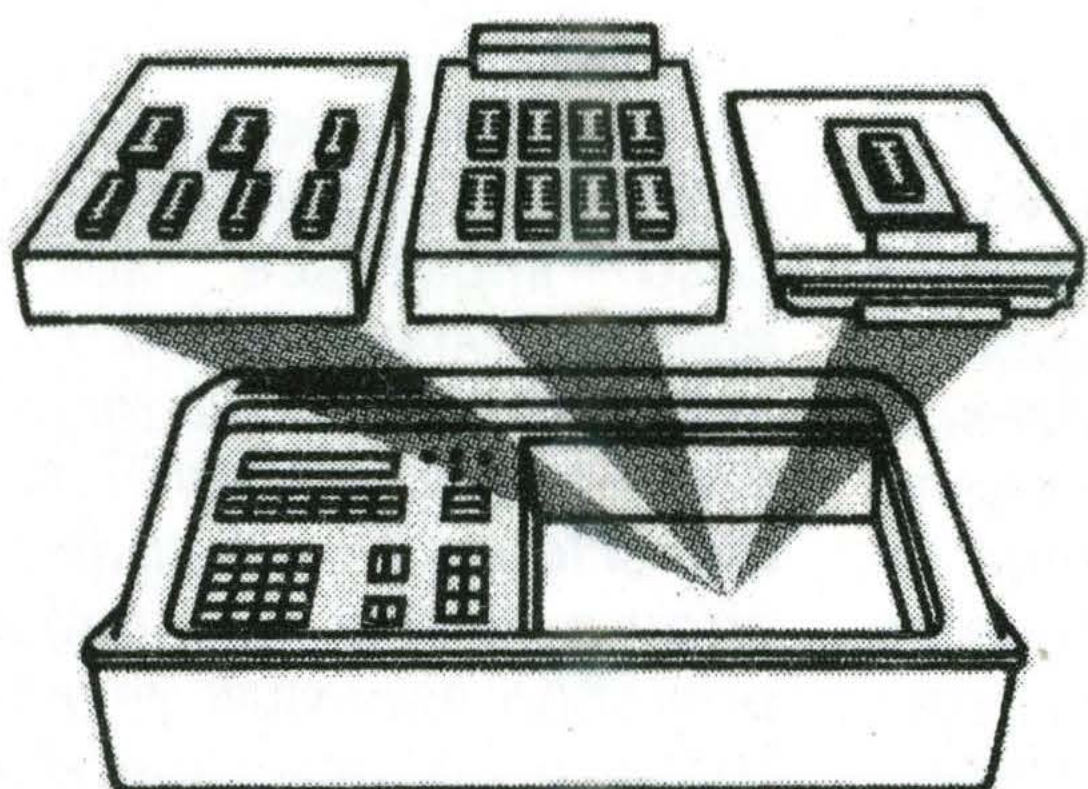
V = 4.1887902 * R ** 3

È inutile farvi notare - spero - che nel calcolo di costanti da dare in pasto al calcolatore è utilissimo il suo fratello minore: il calcolatore tascabile, obbligatoriamente di tipo scientifico e - ancora meglio - programmabile anche lui.
E - come al solito - ancora un commento: notate che *non* ho usato finora le parentesi. ciò è dovuto al fatto che un'analisi delle espressioni in BASIC ci dice subito che le operazioni vengono eseguite dal calcoltatore proprio nel modo che noi desideriamo.
Prendiamo per esempio proprio il volume della sfera: anzitutto il

IL SISTEMA DATA I/O 19 ORA PROGRAMMA PIÚ DI 200 PROM DIVERSE SELEZIONABILI VIA SOFTWARE CON UN SOLO MODULO

Il nostro nuovo Sistema 19/UNIPAK ti permette di programmare la maggior parte delle PROM sia MOS che bipolari da AMD, FAIRCHILD, HARRIS, INTEL, MMI, MOTOROLA, NATIONAL, RAYTHEON, SIGNETICS, e TEXAS INSTRUMENT.
Il software dell'UNIPAK compila l'algoritmo di programmazione e seleziona lo zoccolo appropriato per le PROM da 16, 18, 20, 24 e 28 piedini.

Il Sistema 19/UNIPAK ti dà maggiore libertà nella progettazione e nell'acquisto.
Ciò significa che puoi scegliere la PROM migliore per ogni applicazione e potrai sempre avere un secondo fornitore in modo da ottenere sempre il prezzo migliore e la consegna più breve.

L'omologazione delle case madri e la facilità di calibrazione aiutano a tenere più alta la resa.
L'Unipak ha avuto l'approvazione scritta dai costruttori di componenti. La facilità di calibrazione ti permette di avere la macchina sempre entro le specifiche dei costruttori di componenti.

Gli algoritmi dell'UNIPAK rendono minimo il tempo di programmazione mettendo in risalto l'utilità del Mod. 19 in produzione.
L'Unipak è il primo modulo che usa un nuovo algoritmo che rende possibile programmare una EPROM di 64K in un tempo minore della metà di quello richiesto da un metodo standard per programmare una EPROM di 16K.
Il Sistema 19/UNIPAK è facile da usare e richiede un tempo minimo di training.

Il nuovo concetto del Sistema 19 è una strada senza fine e resterà sempre il migliore.
Sistema 19 è progettato attorno ad un main-frame ed a moduli plug-in. I moduli ora disponibili includono L'UNIPAK, un modulo gang per 8 componenti MOS ed una serie di moduli di programmazione per componenti logici e per singole famiglie di PROM.

I 23 formati di comunicazione ne includono 6 per sistemi di sviluppo.
Sistemi di sviluppo, calcolatori, teletype e terminali CRT sono interfacciabili facilmente con il Sistema 19. Il sistema 19 accetta istruzioni da microprocessori secondo diversi codici da Fairchild, Intel, Millennium, Motorola, Tektronix ed altri emulatori o sistemi di sviluppo senza dover utilizzare interfaccie speciali.

Permettici di mostrarti il futuro
Il nuovo DATA I/O Sistema 19/UNIPAK è ora disponibile. Per una demo o per ricevere documentazione tecnica dettagliata telefona o scrivi alla SISTREL.

DATA I/O

SISTREL

Via Pelizza da Volpedo 59 - 20092 CINISELLO B.
tel. (02) 6181893 - Telex 320346
...ppe Armellini 39, 00143 ROMA. Tel. (06) 5915553 - Telex 680356
...Parco S. Paolo 35, 80126 NAPOLI - Tel. (081) 7679700

☐ Ricevere un'offerta ☐ La visita di un Vs. Tecnico ☐ Essere inseriti nel Vs. mailing list.

COGNOME ....

CITTÀ .... TEL. ....

Il calcolatore per i giovani

calcolatore esegue l'operazione più alta in gerarchia, cioè fa il cubo del raggio. Poi si trova con una serie di operazioni di *uguale gerarchia*. Non vi ho detto cosa fa in questo caso però la risposta è evidente: comincia ad eseguire le operazioni in sequenza, partendo da sinistra e procedendo verso destra. Non vorrei sembrare pedante: ci sono dei linguaggi in cui si segue esattamente la convenzione opposta!
Tornando al nostro volume della sfera, dopo aver effettuato il calcolo del cubo del raggio il calcolatore farà la divisione 4/3 (ed otterrà 1.3333...) ed il risultato verrà moltiplicato alla fine per il cubo del raggio; ed il risultato sarà evidentemente proprio il volume della sfera.

Area di una corona circolare

Se R (che in BASIC chiameremo R1) è il raggio più grande ed r (che in BASIC chiameremo R2) è il raggio più piccolo la formula è:

$$S = \pi (R^2 - r^2)$$

Ecco un caso in cui dobbiamo usare le parentesi! Ed infatti scriveremo:

A = 3.1415927 * (R1 ** 2 - R2 ** 2)

per assicurarci la corretta sequenza delle operazioni.
Eccovi tre domande cattive:
— Se nell'espressione precedente *toglieste le parentesi*, a qual formula corrisponderebbe l'espressione in BASIC risultante?
— Se volete scrivere in BASIC l'espressione dell'area della corona circolare *senza usare le parentesi*, come dovreste fare?
— Restando nel caso precedente pensate che scrivendo la formula in questo modo il calcolatore impiegherebbe *più o meno tempo* di quanto ne impieghi eseguendo l'espressione così come l'abbiamo scritta all'inizio?

Inoltre vi do un altro "compito per casa", che ritengo però piuttosto divertente: andate a scovare delle formule nei vostri libri di scuola (non importa quali siano, purchè semplici e puramente algebriche per il momento) ed esercitatevi a tradurle in BASIC. Ricordatevi sempre dei seguenti punti:
— di come si scrivono i numeri (che si chiamano *le costanti*, in linguaggio tecnicamente corretto);
— del fatto che le variabili potete indicarle con *una* lettera o al massimo con una lettera seguita da un numero. Se questa restrizione vi sembra pesante (ed avete ragione!) allora dovete rivolgervi a linguaggi più elaborati del BASIC. Magari al Pascal .....;
— della gerarchia delle operazioni. Cosa, questa, *molto* importante;
— tutte le istruzioni del BASIC devono essere numerate.

Alla prossima puntata; e buon lavoro. ■

Soluzioni degli esercizi presentati nel numero precedente (BIT N° 7)

Al solito ecco le istruzioni dei problemi proposti nel numero scorso: vi ricordo che si trattava di eseguire alcune somme (e sottrazioni) proposte nel sistema decimale, dando le soluzioni nel sistema esadecimale (HD) e tenendo conto del fatto che i numeri negativi devono venir epressi nella notazione del "complemento a due" spiegata nell'articolo. Eccovi le operazioni proposte (nel sistema decimale) e le soluzioni che avreste dovuto trovare:

Operazioni	**Soluzioni**
65 - 128	*= FF C0*
1234 + 735	*= 07 B1*
12856 - 31745	*= B6 36*
17267 + 22456	*= 9B 2B* che è maggiore di 7F FF (l'operazione non è consentita)
6547 - 17722	*= ED 82*
-1378 - 1255	*= F5 B6*

Voglio spendere però alcune parole sulla quarta operazione, che ho sbrigativamente liquidato dicendovi che il risultato non è compatibile con i limiti imposti ai numeri interi: infatti il più grande numero positivo che si può rappresentare in una parola a 16 bits è 7F FF, e 9B 2B è ovviamente più grande di questo numero. Questa mia affermazione è però vera solo a metà, dal momento che parecchi di voi si saranno accorti che 9B 2B è si un numero maggiore di 7F FF, ma che è pur sempre un numero che rientra nella parola di 16 bits, e quindi in quanto tale è un qualcosa di "legale", cioè deve rappresentare pur sempre qualcosa che il calcolatore capisce.

In effetti è così. Se provate a scrivere per esteso questo numero, scoprirete l'arcano:

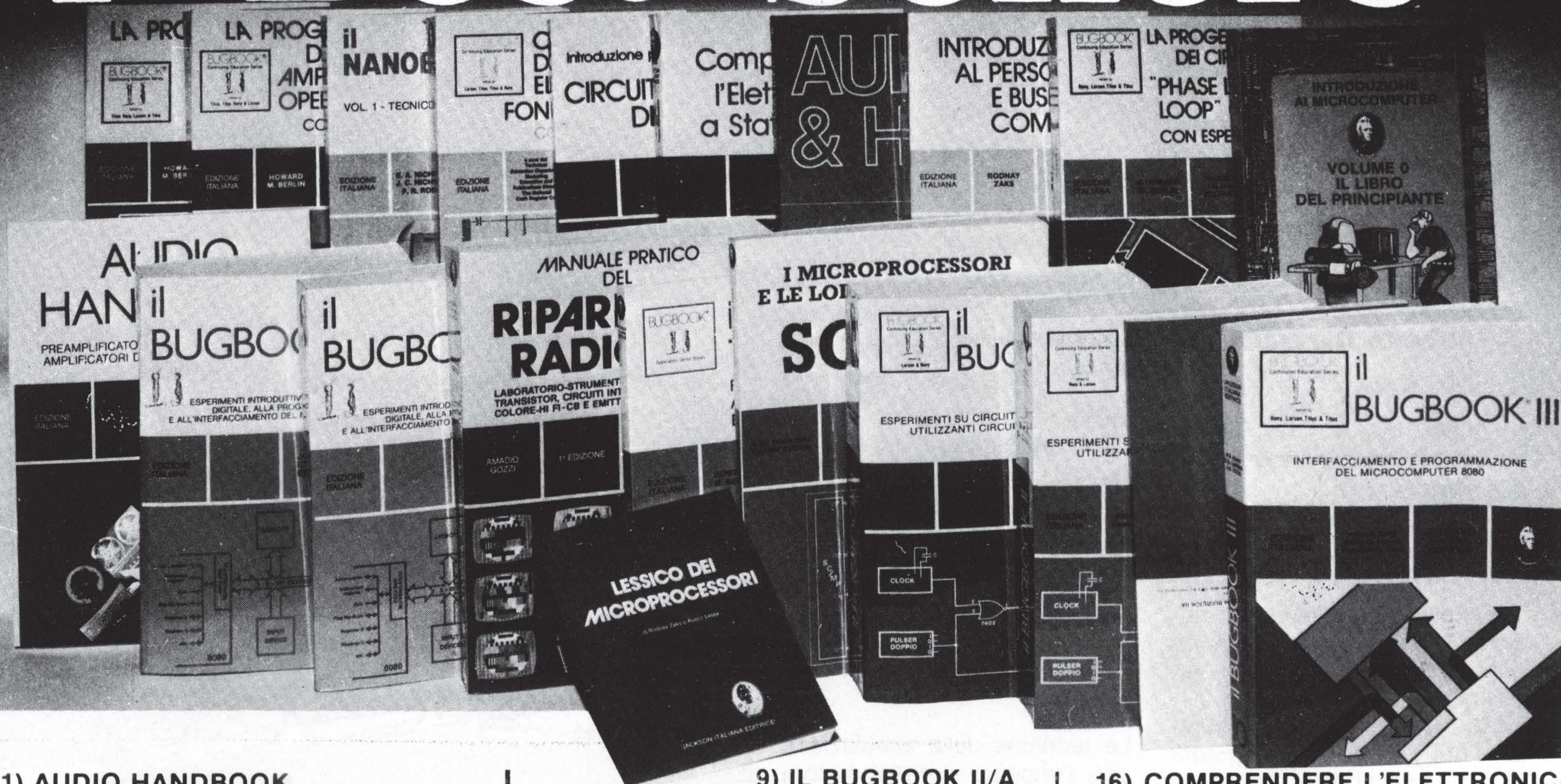
9B 2B = 1001 1011 0010 1011

E l'arcano consiste nel fatto che in questo numero (che dovrebbe essere un numero positivo, nelle nostre intenzioni!) il bit 15 è 1, e ciò sappiamo essere l'indicazione del fatto che siamo in presenza di un numero che il calcolatore capirà come negativo. *Più precisamente 9B 2B verrà capito come il complemento a due di 64 B4 (provate con carta e matita).*
Ed il guaio è che non esiste alcun mezzo per far capire al calcolatore che le cose stanno in maniera diversa. Ancora più chiaramente: 9B 2B è la rappresentazione in complemento a due del numero decimale -25780, cosicchè l'operazione proposta la volta scorsa verrebbe in realtà svolta dal calcolatore in ogni caso, *ma con un risultato a dir poco imbarazzante, e cioè:*

17267 + 22456 = -25780

che evidentemente è sbagliato.

C'è modo di difendersi da insidie di questo genere? Il calcolatore sbaglia? È impazzito il Cervello Elettronico e la Nostra Astronave verrà persa negli Spazi Interstellari? Tranquillizzatevi: niente di così romanzesco. Più semplicemente dovete sempre *tener presente, quando lavorate, o lavorerete, con numeri interi, che il calcolatore li gestisce* nel loro intervallo di definizione, *e che uscire da questo intervallo comporta degli incontrollabili rischi, tipo quello or ora proposto. l'unica difesa contro queste insidie consiste in* controlli *inseriti* nel programma in punti strategici *che avvertano con una segnalazione di errore quando un numero intero si appresta ad uscire dai limiti a lui concessi.*
Spesso infatti non basta essere sicuri di aver scritto correttamente un programma, nè vale prendersela col calcolatore se si ottengono risultati folli. In realtà il mestiere di chi programma è spesso estremamente frustante, dal momento che ogni errore o malfunzionamento è regolarmente e sempre ascrivibile solo a chi ha steso il programma o a chi lo sta usando impropriamente, e praticamente mai alla macchina stessa. Può anche accadere che il calcolatore sbagli, ma in questo caso "impazzisce" solo per qualche romanziere: la realtà è molto più banale, dal momento che in questi rari casi il calcolatore si ferma. In tutti gli altri casi errori o risultati divertenti sono purtroppo ascrivibili esclusivamente a chi usa il calcolatore stesso.

i "Best-Sellers"

1) AUDIO HANDBOOK
Manuale di progettazione audio con progetti completi.
L. 9.550 (Abb. L. 8.550)

2) IL BUGBOOK V
Esperimenti introduttivi all'elettronica digitale alla programmazione e all'interfacciamento del microprocessore 8080 A.
L. 19.000 (Abb. L. 17.100)

3) IL BUGBOOK VI
Completa la trattazione del Bugbook V.
L. 19.000 (Abb. L. 17.100)

4) MANUALE PRATICO DEL RIPARATORE RADIO-TV
Il libro scritto da un riparatore per i riparatori.
L. 18.500 (Abb. L. 16.650)

5) IL TIMER 555
Oltre 100 circuiti pratici e numerosi esperimenti.
L. 8.600 (Abb. L. 7.740)

6) SC/MP
Applicazioni e programmi sul microprocessore SC/MP.
L. 9.500 (Abb. L. 8.550)

7) IL BUGBOOK I
Esperimenti su circuiti logici e di memoria utilizzanti circuiti integrati TTL.
L. 18.000 (Abb. L. 16.200)

8) IL BUGBOOK II
Completa la trattazione del Bugbook I.
L. 18.000 (Abb. L. 16.200)

9) IL BUGBOOK II/A
Esperimenti di interfacciamento e trasmissione dati utilizzanti il ricevitore trasmettitore universale asincrono (UART) e il Loop di corrente a 20 mA.
L. 4.500 (Abb. L. 4.050)

10) IL BUGBOOK III
Interfacciamento e programmazione del microcomputer 8080 A.
L. 19.000 (Abb. L. 17.100)

11) LA PROGETTAZIONE DEI FILTRI ATTIVI
Tutto ciò che è necessario sapere sui filtri attivi.
L. 15.000 (Abb. L. 13.500)

12) LA PROGETTAZIONE DEGLI AMPLIFICATORI OPERAZIONALI
Tutto ciò che è necessario sapere sugli OP-AMP.
L. 15.000 (Abb. L. 13.500)

13) IL NANOBOOK - 80 - VOL. 1
Tecniche di programmazione.
L. 15.000 (Abb. L. 13.500)

14) CORSO DI ELETTRONICA FONDAMENTALE
Testo ormai adottato nelle scuole per il suo alto valore didattico. Per capire finalmente l'elettronica dalla teoria atomica ai circuiti integrati attraverso una esposizione comprensibile a tutti. Esperimenti e test completano la trattazione.
L. 15.000 (Abb. L. 13.500)

15) INTRODUZIONE PRATICA ALL'IMPIEGO DEI CI DIGITALI
Consente un rapido apprendimento dei circuiti integrati.
L. 7.000 (Abb. L. 6.300)

16) COMPRENDERE L'ELETTRONICA A STATO SOLIDO
Un corso autodidattico in 12 lezioni per comprendere tutti i semiconduttori e come questi funzionano insieme in sistemi elettronici.
L. 14.000 (Abb. L. 12.600)

17) AUDIO & HI-FI
Una preziosa guida per chi vuole conoscere tutto sull'hi-fi
L. 6.000 (Abb. L. 5.400)

18) INTRODUZIONE AL PERSONAL & BUSINESS COMPUTING
Un'introduzione esauriente e semplice al mondo affascinante del microcomputer.
L. 14.000 (Abb. L. 12.600)

19) LA PROGETTAZIONE DEI CIRCUITI PLL
Tutto ciò che è necessario sapere sui circuiti "Phase Locked Loop" (PLL).
L. 14.000 (Abb. L. 12.600)

20) INTRODUZIONE AI MICROCOMPUTER VOL. 0 IL LIBRO DEL PRINCIPIANTE
Un corso per coloro che non sanno niente (o quasi) sui calcolatori e gli elaboratori.
L. 14.000 (Abb. L. 12.600)

21) LESSICO DEI MICROPROCESSORI
Un pratico riferimento a tutti coloro che lavorano nel campo dei microcalcolatori o che ad esso sono interessati.
L. 3.500 (L. 3.150)

CEDOLA DI COMMISSIONE LIBRARIA

da inviare a Jackson Italiana Editrice srl - Piazzale Massari, 22 - 20125 Milano

Nome ____________

Cognome ____________

Via ____________ N. ______

Città ____________ Cap ______

Codice Fiscale (indispensabile per le aziende) ____________

Data ______ Firma ____________

SCONTO 10% AGLI ABBONATI

Inviatemi i seguenti volumi:

☐ Pagherò al postino l'importo indicato più spese di spedizione

☐ Allego assegno n° ____________

di L. ____________ (in questo caso la spedizione è gratuita)

☐ **Abbonato** ☐ **Non abbonato**
Barrare i numeri che interessano

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

di **Umberto Barzaghi**

La simulazione mediante elaboratori elettronici

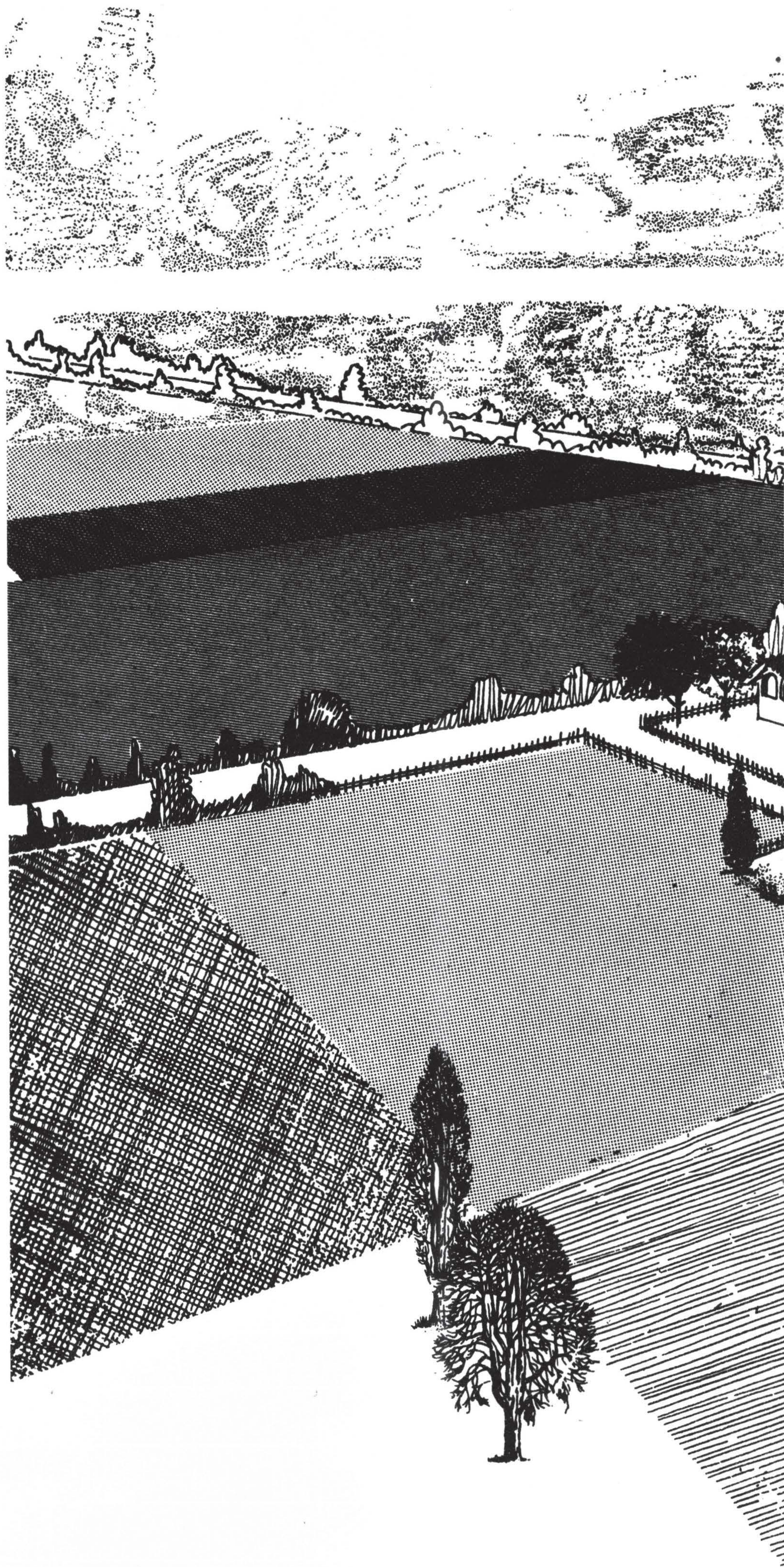
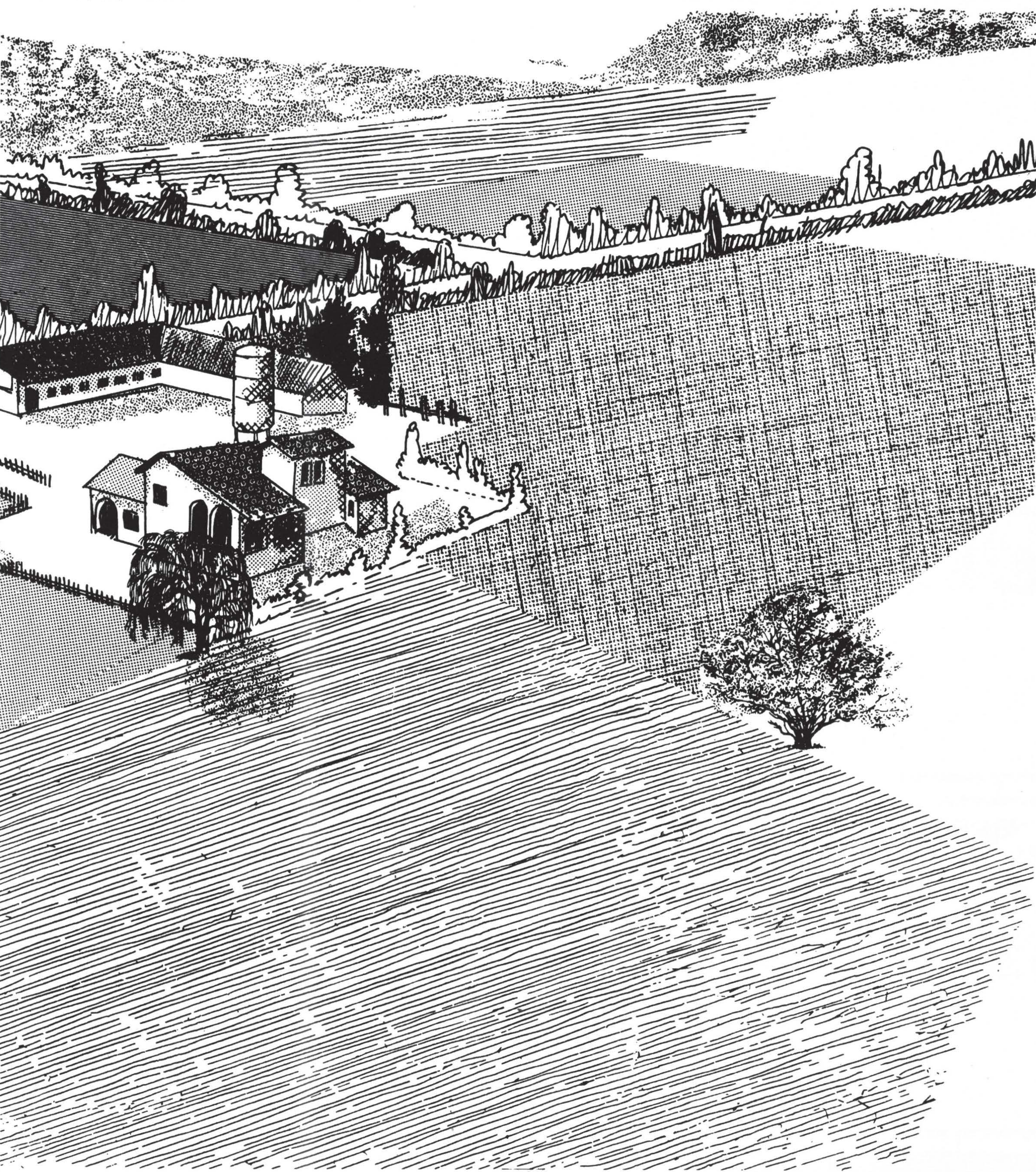
La simulazione mediante calcolatore (*machine or computer simulation*) ha le sue radici nella matematica.
La teoria delle probabilità, la teoria dei giochi (brillantemente e, a quanto si dice, fruttuosamente applicata da Pascal e Cartesio ai giochi d'azzardo in voga ai loro tempi) vengono impiegate per inserire elementi fortuiti in sequenza casuale nelle attività simulate.
Lo scopo della simulazione tramite calcolatore è duplice: trovare le risposte e, tramite la loro analisi, capire i procedimenti. infatti, l'insieme dei dati ottenuti, dei fattori casuali e delle restrizioni programmate nel calcolatore fanno si che la partecipazione umana non si limiti allo sviluppo iniziale del programma, ma cerchi di comprendere i risultati ottenuti. A questo punto si entra in un confronto con la propria "creatura" nel quale la perspicacia dell'utente, alimentata dall'interesse della simulazione, è messa a dura prova dalla difficoltà di implementazione della stessa. L'elaboratore, infatti, tratta un grande numero di variabili, ben al di là di quello che il cervello umano può fare, per cui la partecipazione effettiva al processo di simulazione è tanto più limitata quanto più esso è complesso.

Le simulazioni più comuni eseguite con l'aiuto dell'elaboratore sono quelle in cui esso stesso agisce da intermediario tra i due giocatori alfine di (1) sveltire il gioco, (2) far fronte alla complessità dei calcoli, (3) assicurare il massimo grado di precisione. La maggior parte delle simulazioni di gestione aziendale e d'economia o di affari è di questo tipo.

Esistono anche casi "uomo contro elaboratore", ed in alcuni casi gli elaboratori possono giocare contro gli elaboratori; si pensi all'annuale campionato di scacchi statunitense nel quale le università gareggiano tra loro per programmare i propri elaboratori in modo che diventino i campioni degli Stati Uniti!

Le tecniche della simulazione con l'impiego dell'elaboratore sono state applicate tanto ad argomenti seri che ad argomenti frivoli. Si sono così simulati incontri di pugilato tra pugili appartenenti ad epoche diverse (recentemente la Gazzetta dello Sport, in collaborazione con la RAI, ha fatto qualcosa di molto simile con un Giro d'Italia fra Merckx, Anquetil, Coppi etc.); si è tentato di prevedere il risultato di avvenimenti, sportivi e non, di grande rilievanza, quali i mondiali di calcio, le elezioni presidenziali americane, etc.
Per quanto riguarda i giochi di simulazione e tutto quanto loro attiene, ho fatto riferimento al volume *I giochi di simulazione* di John L. Taylor e Rex Walford, edito dalla University Press. Ad uno degli esempi pratici di simulazione da loro riportati nel suddetto volume è ispirato il gioco che segue.

Il gioco della Fattoria dell'Herefordshire

La "Fattoria dell'Herefordshire" è un gioco relativamente semplice. Probabilmente giocarlo non dà grandi "soddisfazioni", ma, e qui forse si capisce che cosa si intende per simulazione, diventa interessante se i giocatori s'immedesimano nella situazione simulata, che nel nostro caso consiste nel cercare di ottenere il massimo reddito dalle colture di una fattoria.

Un gioco di simulazione: la fattoria di Mister Brown

Il primo passo consiste pertanto nel considerare gli elementi che influiscono sulla resa dei vari tipi di coltivazione, e cioè, ad esempio, il clima, la fertilità del terreno, il tipo di coltura, etc. Già in questa prima fase appare chiaro il fine didattico del gioco (che non a caso è stato adottato nelle scuole secondarie inglesi); grazie ad esso infatti si imparano alcune nozioni di agricoltura, con le relative implicazioni pratiche.
Il secondo passo consiste nell'implementare il problema su calcolatore; alcuni dati sono ben noti, mentre altri, non esattamente valutabili, si sottopongono a limitazioni approssimative o si fanno intervenire come elementi casuali. A questo punto perciò il problema può essere riportato ad un algoritmo riproducibile per il calcolatore.
Nel nostro caso le limitazioni che intervengono saranno esposte più avanti.
La terza fase consiste naturalmente nell'analisi dei risultati ottenuti, analisi che permette da un lato di scoprire soluzioni altrimenti non individuabili, dall'altro di definire un modello matematico più aderente al problema reale iniziale. Chiaramente noi non affrontiamo qui questa fase; invitiamo chi fosse interessato al problema a scriverci.

Così come è stato implementato, il problema fa riferimento ad una particolare configurazione regionale; una variante potrebbe essere quella di considerare ad esempio una zona tropicale aventi ritmi stagionali più marcati e prevedibili, così come è possibile considerare altre zone con andamenti climatici più complessi. Nel nostro caso i tipi di clima sono stati ovviamente semplificati, ma assomigliano alle stagioni inglesi, e lo stesso si può dire della fertilità del suolo.
Inoltre, dato che la rotazione tradizionale delle colture non è pratica universalmente diffusa,

la fattoria di Mister Brown

non è stata implementata nel programma (ma calorosamente raccomandata!).
Scendendo ulteriormente nei particolari dell'implementazione del gioco, precisiamo che:
— le coltivazioni possono essere a breve e a lungo termine (ecco il perchè del carattere permanente attribuito ad alcune colture e del mancato reddito dal bosco durante il periodo del gioco);
— non solo il clima agisce sulla resa della colture, ma esiste un elemento fortuito che sfugge all'"agricoltore" (la funzione RND del linguaggio BASIC, generatrice di numeri casuali).
I risultati che si ottengono da questa implementazione sono comunque abbastanza interessanti: fra l'altro è possibile dimostrare come l'uomo possa decidere l'uso del terreno, salvo che in luoghi ove esistano condizioni fisiche radicali.

Numero dell'appezzamento	Area (in ettari)	Tipo di coltura
1	10	pascolo permanente
2	14	libera scelta
3	12	libera scelta
4	3	ribes
5	14	libera scelta
6	4	luppolo
7	8	libera scelta
8	10	libera scelta
9	4	mele da sidro
10	4	bosco
11	5	libera scelta
12	6	libera scelta
Totale	94	

TABELLA I - Questa tabella definisce sia l'area di ogni appezzamento di terreno che la possibile coltura seminabile.

Natura della coltivazione	Clima			
	Caldo piovoso	Caldo asciutto	Fresco piovoso	Fresco asciutto
Orzo	7	10	7	10
Frumento	7	15	5	5
Avena	7	5	10	5
Piselli	20	7	2	2
Fagioli	10	5	5	2
Barbabietola da zucchero	22	12	7	5

TABELLA II - Prima tabella dei redditi. In conformità al clima, che nel programma è previsto quale elemento casuale, per ogni coltura viene definito il relativo reddito per ettaro.

Natura della coltivazione	Clima			
	Caldo piovoso	Caldo asciutto	Fresco piovoso	Fresco asciutto
Luppolo	30	37	22	22
Mele da sidro	37	37	25	10
Ribes	30	49	15	20
Pascolo permanente	10	5	10	2
Non si ottengono redditi dal bosco				

TABELLA III - Come la tabella II definisce il reddito per ettaro per le colture prefissate.

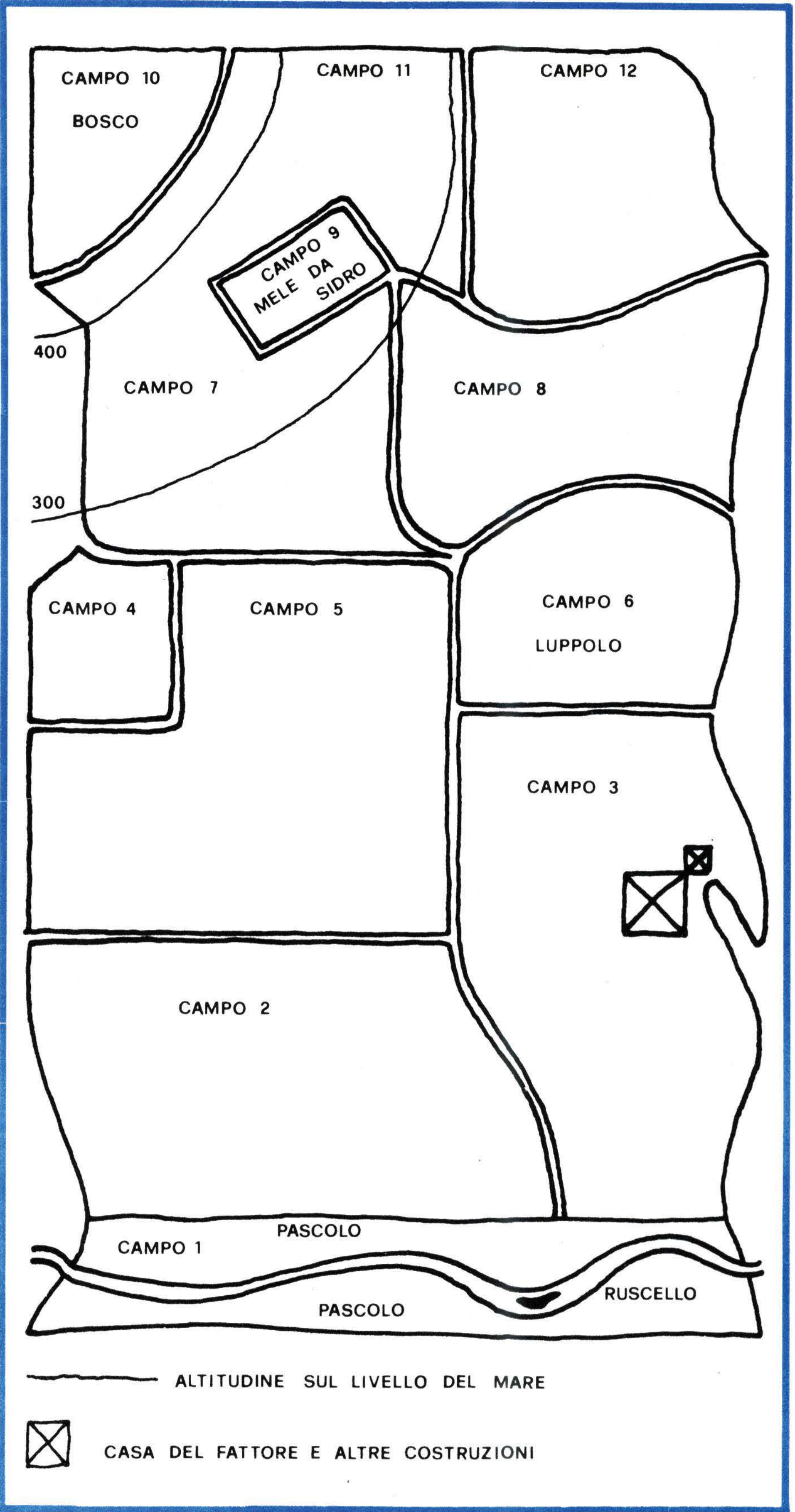



Figura 1 - La mappa della fattoria dell'Herefordshire.

Ulteriori dati

Scopo del gioco è trovare le colture che danno il massimo reddito, senza però cambiare colture preesistenti, ma intervenendo solo sugli appezzamenti di terreno liberi.

In particolare occorre sapere che:

— la fattoria occupa circa 94 ettari ed è divisa in 12 appezzamenti (i dati relativi agli appezzamenti sono riportati nella *Tabella I*);
— ogni anno si può coltivare uno qualunque dei sei prodotti elencati nella *Tabella II* nei campi a libera scelta. È consigliabile tuttavia, per aumentare il realismo della simulazione, non coltivare lo stesso prodotto in due campi contigui o per due anni conseguitivi nello stesso campo, perchè ciò impoverirebbe il terreno;
— la resa delle colture, presumendo che la fertilità dei campi sia uniforme, dipenderà dalle condizioni atmosferiche nel corso dell'anno (il relativo reddito si può ricavare dalla *Tabella II*). Si può avere uno dei seguenti tipi di clima: caldo piovoso, caldo asciutto, fresco piovoso, fresco asciutto. Dato che non si sa quante volte e quando ciascun tipo di tempo si verificherà, sarà il calcolatore a stabilirlo mediante la generazione di numeri casuali;
— le colture elencate in *Tabella II* possono essere fatte su

REMarks

Riga 15:	*Viene azzerata, a inizio programma, la variabile* RT = *reddito totale.*
Riga 20:	*Viene stampata una richiesta di introduzione dati riguardante il numero di anni che si intende prendere in considerazione; si consigliano giocate massime di 5 anni (periodi superiori impediscono di seguire adeguatamente lo sviluppo del gioco e di conprendere appieno le finalità).*
Riga 25:	*La variabile* N *introdotta (numero di anni considerati) viene usata come limite superiore del più esterno dei due cicli FOR anni/dati che regolano lo sviluppo del programma; l'indice di questo primo ciclo è uguale ad 1.*
Riga 35:	*La funzione RND generatrice di numeri casuali provvede all'attribuzione del clima.*
Righe 40-70:	*La ripartizione delle probabilità di ricorrenza dei quattro tipi di clima è in questa implementazione della simulazione assolutamente equa (25% ciascuno). Questo, fatto dà, forse, un'idea della variabilità delle condizioni atmosferiche in Inghilterra. Una diversa ripartizione delle probabilità tra i quattro tipi di clima sarebbe, comunque, agevolmente ottenibile sostituendo nelle righe 40, 45 e 50 ai valori 24, 49 e 74 altri valori (ad esempio 9, 49 e 79 per ottenere 10% di caldo piovoso, 40% di caldo asciutto, 30% di fresco piovoso e 20% di fresco asciutto).*
Riga 72:	*Vengono azzerate a inizio ciclo le variabili* RP = *reddito parziale e* RA = *reddito annuo.*
Riga 75:	*Viene iniziato un ciclo più interno (di indice J) che provvede a richiedere l'introduzione del tipo di coltura prescelta per i vari campi, saltando opportunamente i campi a scelta obbligata; gli indici che contraddistinguono i diversi tipi di coltivazione vengono presentati ogni volta al giocatore insieme alla richiesta di introduzione della scelta effettuata.*
Righe 80-95, 100-115, 120-135, 140-155 e 160-170:	*Provvedono a calcolare il reddito parziale dei campi a coltivazione prefissata a seconda dei diversi tipi di clima (tranne che per il campo N. 10, coltivato a bosco, per il quale viene stampato un opportuno avviso) e ad aggiornare il reddito annuo.*
Righe 185-225:	*Realizzano quanto sopra descritto per i campi a libera scelta; in particolare:*
Righe 185-188:	*richiedono l'introduzione della scelta circa il prodotto da coltivare;*
Righe 190-195:	*associano al prodotto prescelto una opportuna stringa contenente il nome del prodotto;*
Righe 200-205:	*associano ad ogni campo le corrispondenti dimensioni come da piantina e da Tabella I;*
Righe 210-218:	*associano al variare dei prodotti prescelti e del clima il reddito per ettaro corrispondente, come da Tabella II;*
Righe 220-221:	*stampano il reddito del J-esimo campo per l'I-esimo anno indicando inoltre il tipo di coltivazione prescelta;*
Riga 225:	*Chiude il più interno dei due cicli FOR anni/dati.*
Riga 230:	*Stampa il reddito per l'I-esimo anno.*
Riga 233:	*Aggiorna il reddito totale.*
Righe 235-240:	*Associano ai vari tipi di clima una stringa contenente il nome e la stampa a seconda del clima dell'I-esimo anno.*
Riga 245:	*Chiude il più esterno dei due cicli FOR anni/dati.*
Riga 250:	*Stampa il reddito totale risultante dalle N giocate.*

la fattoria di Mister Brown

```
GIOCO DELLA FATTORIA

10 PRINT "FATTORIA DEL SIGNOR HUNT"
11 PRINT
12 PRINT
15 RT=0
20 PRINT "QUANTI ANNI TI INTERESSANO"
21 PRINT :PRINT
25 INPUT N
30 FOR I=1 TO N
35 LC=RND(I)*100
40 IF LC<24 THEN 60
45 IF LC<49 THEN 65
50 IF LC<74 THEN 70
55 CL=4
56 GOTO 71
60 CL=1
61 GOTO 71
65 CL=2
66 GOTO 71
70 CL=3
71 TIME$="000000"
72 RP=0 : RA=0
75 FOR J=1 TO 12
80 IF J<>1 THEN 100
85 H=10
86 PRINT "APPEZZAMENTO N.1 COLTIVATO A PASCOLO" : PRINT : PRINT
90 IF CL=1 OR CL=3 THEN R=10
91 IF CL=2 THEN R=5
92 IF CL=4 THEN R=2
93 RP=H*R : RA=RA+RP
95 PRINT "IL REDDITO DEL CAMPO N.1 PER IL ";I; "O ANNO E' DI"; RP:PRINT:PRINT
96 GOTO 225
100 IF J<>4 THEN 120
105 H=3
106 PRINT "APPEZZAMENTO N.4 COLTIVATO A RIBES" : PRINT : PRINT
110 IF CL=1 THEN R=30
111 IF CL=2 THEN R=49
112 IF CL=3 THEN R=15
113 IF CL=4 THEN R=20
114 RP=H*R:RA=RA+RP
115 PRINT "IL REDDITO DEL CAMPO N.4 PER IL "; I; "O ANNO E' DI";RP:PRINT:PRINT
116 GOTO 225
120 IF J<>6 THEN 140
125 H=4
126 PRINT "APPEZZAMENTO N.6 COLTIVATO A LUPPOLO" : PRINT : PRINT
130 IF CL=1 THEN R=30
131 IF CL=2 THEN R=37
132 IF CL=3 OR CL=4 THEN R=22
133 RP=H*R : RA=RA+RP
135 PRINT "IL REDDITO DEL CAMPO N.6 PER IL "; I; "O ANNO E' DI";RP:PRINT:PRINT
136 GOTO 225
140 IF J<>9 THEN 160
145 H=4
146 PRINT "APPEZZAMENTO N.9 COLTIVATO A MELE" : PRINT : PRINT
150 IF CL=1 OR CL=2 THEN R=37
151 IF CL=3 THEN R=25
152 IF CL=4 THEN R=10
153 RP=H*R : RA=RA+RP

155 PRINT "IL REDDITO DEL CAMPO N.9 PER IL "; I; "O ANNO E' DI";RP:PRINT:PRINT
156 GOTO 225
160 IF J<>10 THEN 180
165 H=4
166 PRINT "APPEZZAMENTO N.10 COLTIVATO A BOSCO" : PRINT :PRINT
170 PRINT "L'APPEZZAMENTO N.10 NON DA' REDDITO" : PRINT: PRINT
171 GOTO 225
180 PRINT TIME$
185 PRINT "CHE PRODOTTO DESIDERI COLTIVARE NEL CAMPO N."; J; "?":PRINT:PRINT
186 PRINT "1=ORZO, 2=FRUMENTO, 3=AVENA,"
187 PRINT "4=PISELLI, 5=FAGIOLI, 6=BARBABIETOLA.": PRINT : PRINT
188 INPUT P
190 IF P=1 THEN N$="ORZO"
191 IF P=2 THEN N$="FRUMENTO"
192 IF P=3 THEN N$="AVENA"
193 IF P=4 THEN N$="PISELLI"
194 IF P=5 THEN N$="FAGIOLI"
195 IF P=6 THEN N$="BARBABIETOLA"
200 IF J=2 OR J=5 THEN H=14
201 IF J=3 THEN H=12
202 IF J=7 THEN H=8
203 IF J=8 THEN H=10
204 IF J=11 THEN H=5
205 IF J=12 THEN H=6
210 IF(CL=1AND(P=1ORP=2ORP=3))OR(CL=2ANDP=4)OR(CL=3AND(P=1ORP=6))THENR=7
211 IF((CL=2ORCL=4)ANDP=1)OR(CL=3ANDP=3)OR(CL=1ANDP=5)THENR=10
213 IF CL=4 AND P=6 THEN R=5
214 IF((CL=3ORCL=4)ANDP=4)OR(CL=4ANDP=5)THENR=2
215 IF CL=2 AND P=2 THEN R=15
216 IF CL=1 AND P=4 THEN R=20
217 IF CL=1 AND P=6 THEN R=22
218 IF CL=2 AND P=6 THEN R=12
219 RP=H*R : RA=RA+RP
220 PRINT "APPEZZAMENTO N."; J;"COLTIVATO A "; N$ : PRINT : PRINT
221 PRINT "IL REDDITO DEL CAMPO N."; J;" PER IL "; I; "O ANNO E' DI "; RP
222 PRINT : PRINT
225 NEXT J
230 PRINT "IL REDDITO ANNUO TOTALE PER IL "; I; "O ANNO E' DI "; RA
231 PRINT : PRINT
233 RT=RA+RT
235 IF CL=1 THEN M$="CALDO PIOVOSO"
236 IF CL=2 THEN M$="CALDO ASCIUTTO"
237 IF CL=4 THEN M$="FRESCO PIOVOSO"
238 IF CL=4 THEN M$="FRESCO ASCIUTTO"
240 PRINT "IL TEMPO PER L'ANNO APPENA TRASCORSO E' STATO "; M$ :PRINT:PRINT
245 NEXT I
250 PRINT"IL REDDITO TOTALE E' DI ";RT
260 END
```

Listing del programma

```
FATTORIA DEL SIGNOR HUNT

QUANTI ANNI TI INTERESSANO      1

APPEZZAMENTO N.1 COLTIVATO A PASCOLO
IL REDDITO DEL CAMPO N.1 PER IL  1 O ANNO E' DI 100
000000
CHE PRODOTTO DESIDERI COLTIVARE NEL CAMPO N. 2 ?   6
1=ORZO, 2=FRUMENTO, 3=AVENA,
4=PISELLI, 5=FAGIOLI, 6=BARBABIETOLA.
APPEZZAMENTO N. 2 COLTIVATO A BARBABIETOLA
IL REDDITO DEL CAMPO N. 2  PER IL  1 O ANNO E' DI  56
000021
CHE PRODOTTO DESIDERI COLTIVARE NEL CAMPO N. 3 ?   3
1=ORZO, 2=FRUMENTO, 3=AVENA,
4=PISELLI, 5=FAGIOLI, 6=BARBABIETOLA.
APPEZZAMENTO N. 3 COLTIVATO A ORZO
IL REDDITO DEL CAMPO N. 3  PER IL  1 O ANNO E' DI  84
APPEZZAMENTO N.4 COLTIVATO A RIBES
IL REDDITO DEL CAMPO N.4 PER IL  1 O ANNO E' DI 45
000034
CHE PRODOTTO DESIDERI COLTIVARE NEL CAMPO N. 5 ?   5
1=ORZO, 2=FRUMENTO, 3=AVENA,
4=PISELLI, 5=FAGIOLI, 6=BARBABIETOLA.
APPEZZAMENTO N. 5 COLTIVATO A FAGIOLI
IL REDDITO DEL CAMPO N. 5  PER IL  1 O ANNO E' DI  210
APPEZZAMENTO N.6 COLTIVATO A LUPPOLO
IL REDDITO DEL CAMPO N.6 PER IL  1 O ANNO E' DI 88
000047
CHE PRODOTTO DESIDERI COLTIVARE NEL CAMPO N. 7 ?   2
1=ORZO, 2=FRUMENTO, 3=AVENA,
4=PISELLI, 5=FAGIOLI, 6=BARBABIETOLA.
APPEZZAMENTO N. 7 COLTIVATO A FRUMENTO
IL REDDITO DEL CAMPO N. 7  PER IL  1 O ANNO E' DI  176
000101
CHE PRODOTTO DESIDERI COLTIVARE NEL CAMPO N. 8 ?   3
1=ORZO, 2=FRUMENTO, 3=AVENA,
4=PISELLI, 5=FAGIOLI, 6=BARBABIETOLA.
APPEZZAMENTO N. 8 COLTIVATO A AVENA
IL REDDITO DEL CAMPO N. 8  PER IL  1 O ANNO E' DI  100
APPEZZAMENTO N.9 COLTIVATO A MELE
IL REDDITO DEL CAMPO N.9 PER IL  1 O ANNO E' DI 100
APPEZZAMENTO N.10 COLTIVATO A BOSCO
L'APPEZZAMENTO N.10 NON DA' REDDITO
000116
CHE PRODOTTO DESIDERI COLTIVARE NEL CAMPO N. 11 ?   6
1=ORZO, 2=FRUMENTO, 3=AVENA,
4=PISELLI, 5=FAGIOLI, 6=BARBABIETOLA.
APPEZZAMENTO N. 11 COLTIVATO A BARBABIETOLA
IL REDDITO DEL CAMPO N. 11  PER IL  1 O ANNO E' DI  55
000127
CHE PRODOTTO DESIDERI COLTIVARE NEL CAMPO N. 12 ?   4
1=ORZO, 2=FRUMENTO, 3=AVENA,
4=PISELLI, 5=FAGIOLI, 6=BARBABIETOLA.
APPEZZAMENTO N. 12 COLTIVATO A PISELLI
IL REDDITO DEL CAMPO N. 12  PER IL  1 O ANNO E' DI  12
IL REDDITO ANNUO TOTALE PER IL  1 O ANNO E' DI  1048
IL TEMPO PER L'ANNO APPENA TRASCORSO E' STATO
IL REDDITO TOTALE E' DI  1048
```

Esecuzione del programma.

uno qualunque dei campi a libera scelta, mentre quelle della *Tabella III* sono limitate unicamente ad appezzamenti prefissati;

— il calcolo del reddito totale è semplice: se si vuole ad esempio coltivare barbabietole da zucchero nel campo N. 2, e il tempo risulta essere caldo asciutto, il reddito del relativo appezzamento è 12 x 14 = 168 cioè reddito per ettaro x numero ettari = reddito appezzamento.

Il reddito totale è la somma dei redditi dei singoli appezzamenti.

Ulteriori particolari sono dati nei commenti del programma che è stato eseguito su PET con istruzioni BASIC standard. . . ■

COMPUTER PRATICO accoglie idee e suggerimenti di utilizzo immediato nell'ambito dell'elaborazione personale. Lo scambio di quanto si è in grado di individuare lavorando con un computer è fondamentale per una diffusione effettiva dell'elaborazione personale. Per promuovere questa iniziativa, ogni testo pubblicato sarà ricompensato da un minimo di L. 50.000 ad un massimo di L. 100.000. I testi devono essere inviati a: Bit Rubrica COMPUTER PRATICO - Piazzale Massari, 22 - 20125 MILANO

Routines

di **Gloriano Rossi**

```
59990 REM********************************
59991 REM******  R O U T I N E S  *****
59992 REM********************************
59998 REM****** PUNTEGGATURA  *********
59999 REM********************************
60000 N$=STR$(N)
60010 LX=LEN(N$)
60020 X$=""
60030 FOR I=LX TO 1 STEP-1
60040 X$=MID$(N$,I,1)+X$
60050 IF I=1 THEN 60090
60060 P=LX-I+1
60070 IF P=3 OR P=6 OR .P=9 OR P=12 THEN X$="."+X$
60080 NEXT
60090 RETURN
60097 REM********************************
60098 REM****** ACCETAZIONE EXAMINE ***
60099 REM********************************
60100 X3$=" "
60110 IF X$="N" THEN X3$="0"
60115 IF X$="Z" THEN X4$=X$
60120 INPUT X$
60130 LX=LEN(X$)
60131 IF X4$="Z" OR X3$="0" THEN 60133
60132 GOTO 60140
60133 FOR I=1 TO LX
60134 A$=MID$(X$,I,1)
60135 IF A$<"0" OR A$>"9" THEN PRINT "NON NUMERICO":GOTO 60120
60136 NEXT
60140 IF LX=X THEN 60250
60150 IF LX>X THEN 60235
60160 N=X-LX
60170 X2$=""
60180 FOR I=1 TO N
60190 X2$=X2$+X3$
60200 NEXT
60210 IF X3$="0" OR X4$="Z" THEN X$=X2$+X$:GOTO 60250
60220 X$=X$+X2$
60230 GOTO 60250
60235 IF X3$="0" OR X4$="Z" THEN X$=RIGHT$(X$,X):GOTO 60250
60240 X$=LEFT$(X$,X)
60250 RETURN
```

ROUTINES

Quante volte abbiamo sentito questa parola?
In questo numero proponiamo due routinette di sicura utilità, la prima delle quali ci permette di ottenere una punteggiatura nei numeri, mentre della seconda, che chiameremo "EXAMINE", vedremo più avanti le non poche specifiche. Prima di vedere come funzionano e che cosa si può in pratica ottenere, è bene spiegare la ragione della nascita di queste routines.

Punteggiatura

In tutti i computers, dal più grande al più piccolo, un numero intero o una variabile numerica (intera) viene trattata così com'è, cioè senza quelle separazioni che noi, esseri umani, usiamo mettere per ragioni di estetica oltre che di praticità di lettura. Un numero quindi appare normalmente composto (da destra verso sinistra) dalle unità, dalle decine, quindi dalle centinaia, poi ancora dalle migliaia, le decine di migliaia e così via senza alcuna punteggiatura o spaziatura che definisca la parte inferiore a mille o la parte specifica delle migliaia o milioni.
Ma, come si suol dire, l'occhio vuole la sua parte, e quindi quando dobbiamo dare un comando di edit sul video del nostro personal computer oppure dobbiamo stampare un tabulato tramite una periferica adeguata, sarebbe bene oltre che bello poter evidenziare il numero in questione con le dovute separazioni.
La prima delle due routines che abbiamo citato soddisfa esattamente l'esigenza estetica della separazione o punteggiatura.
Vediamo in dettaglio come funziona.
L'entrata della routine vede il numero posto nella variabile chiamata N; il risultato sarà posto nella variabile alfanumerica X$.
Dopo aver trasformato il numero N in stringa alfanumerica N$, con il comando BASIC LEN si definisce la lunghezza fisica del numero, ovvero l'occupazione in bytes in memoria. Questa informazione ci servirà, dopo aver annulato la variabile X$ da ogni precedente contenuto, per il trattamento del numero di input.
Inizia a questo punto la trasformazione vera e propria. Innanzitutto la stringa N$ passa ogni carattere (numero) alla stringa di uscita X$ tramite loop definito con il comando FOR: il valore iniziale dell'indice di loop è uguale alla lunghezza fisica di N$, e man mano è decrementato fino a completare il trasferimento. Inoltre è fatto il test di I uguale ad 1 per uscire dal loop prima che questo finisca per suo conto, dato che in caso contrario si otterebbe una punteggiatura superflua, a meno che non si abbia la coincidenza con le centinaia o migliaia .
L'inserimento della punteggiatura viene determinato in primo luogo calcolando il valore della variabile P, poi eseguendo il test di questa variabile con valori prefissati. Se il valore di P corrisponde ad uno di quelli definiti, allora è giunto il momento di aggiungere il punto sulla sinistra della stringa X$ fino ad ora costruita.
L'utilizzazione di questa routine può avvenire in due modi similari:

— se la variabile da trattare è numerica, si esegue:

N=variabile: GOSUB 60000:
PRINT X$

— se la variabile da trattare risulta essere già alfanumerica si esegue:

N = variabile alfanumerica:
GOSUB 60010: PRINT X$

EXAMINE

Per vari motivi di programmazione, molte volte risulta necessario ottenere una determinata variabile alfanumerica di lunghezza predeterminata. Sappiamo d'altro canto che la prerogativa del BASIC è proprio quella che una variabile occupa tanti bytes quanti sono quelli effettivamente occupati dal suo argomento. Infatti se X$ contiene PIPPO e A$ contiene GIUSEPPINA, la variabile X$ occupa 5 bytes, mentre A$ ne occupa 10. Ora se noi volessimo, per una qualsiasi ragione, ottenere sempre variabili di lunghezza

10, non ci sarebbe alcun problema per A$, mentre dovremmo ingegnarci in qualche modo per ciò che concerne X$.
Un altro caso: abbiamo tante variabili alfanumeriche dal contenuto numerico (per esempio A$ = "32", B$ = "1343", C$ = "724") e vogliamo fare in modo che in caso di stampa o di edit su video queste variabili vengano riportate in modo tale che le unità risultino incollonnate con le unità, le decine con le decine, e così via. Vogliamo anche che tutte queste variabili siano lunghe 6 caratteri e che le cifre non significative siano poste a destra del numero e costituite da zeri o spazi a seconda delle necessità.
Sembra questo un caso particolarmente difficile, ma in realtà con una sola routine possiamo ottenere, a seconda dei casi e delle necessità:

- la trasformazione di lunghezza X di una variabile alfanumerica dal contenuto alfanumerico, con allineamento a sinistra e spazi a destra, o troncamento della parte destra per una eventuale eccedenza;
- la trasformazione di lunghezza X di una variabile alfanumerica dal contenuto numerico, con allineamento a destra e zeri / spazi al posto della parte non significativa, o troncamento della parte sinistra per eventuale eccedenza;
- l'accettazione (INPUT) di una stringa alfanumerica dal contenuto alfanumerico, e trattamento come per il primo caso;
- l'accettazione (INPUT) di una stringa alfanumerica dal contenuto numerico, controllo di numericità, e trattamento come per il secondo caso.

Al contrario di quanto abbiamo fatto con la routine di punteggiatura, vediamo prima come è possibile richiamare questa routine al fine di ottenere la risoluzione dei quattro casi citati.
Primo caso: si inizializza la variabile X$ con il contenuto della variabile in questione VAR$, poi si definisce il valore di X (lunghezza che si vuole ottenere), quindi si chiama la routine e all'uscita si ritorna alla variabile VAR$, se necessario, il nuovo contenuto riveduto:

X$ = VAR$: X = lunghezza: GOSUB 60130: VAR$ X $

Secondo caso: si trasforma la variabile di input in stringa alfanumerica, poi si definisce il valore della lunghezza, quindi si richiama la routine dopo aver inizializzato la variabile X3$ con 0 e si ritorna con il contenuto trasformato dalla solita X$:

X$ = STR$ (VAR): X= lunghezza: X3$ = "0": GOSUB 60130: VAR$ = X$

Un'altra soluzione prevede la possibilità di porre la parte non significativa a spazio invece che a zero:

X$ = STR$ (VAR): X= lunghezza: X4$ = "Z": GOSUB 60130: VAR$ =X$

La soluzione del terzo caso risulta alquanto semplice: si deve prevedere la sola lunghezza che si vuole ottenere e si preleva l'informazione sempre dalla variabile X$:

X= lunghezza: GOSUB 60100: VAR$ =X$

Nel quarto caso si prevedono due soluzioni: la prima indica alla routine che il contenuto è numerico, mentre la seconda indica in più la sostituzione degli zeri con spazio nella parte non significativa:

X = lunghezza: X$ "N": GOSUB 60100: VAR$ = X$

oppure

X = lunghezza: X$ = "Z": GOSUB 60100: VAR$ = X$. ■

Remark di EXAMINE

Le variabili di ingresso e di uscita sono:

X	: Lunghezza del campo che si vuole ottenere.
X $	: In input deve contenere la lettera N o la lettera Z se il campo di uscita dovrà essere numerico; in caso contrario la variabile potrà non esistere o contenere qualsiasi altra cosa. In output contiene il risultato (campo di lunghezza X) dalla routine.

Le variabili utilizzate all'interno della EXAMINE sono:

X3 $	: Il contenuto è uno spazio oppure uno zero.
X4 $	: Avrà come contenuto X$ di ingresso se questo è stato inizializzato con la lettera Z.
A$	: Variabile di test di contenuto numerico.
LX	: Lunghezza della stringa di input da trattare.
X2 $	: Stringa di lunghezza variabile da un minimo di zero bytes ad un massimo di X-1, che sarà unita in testa o in coda alla X$ di uscita.
N	: Numero corrispondente alla lunghezza della variabile da costruire a zero/spazio al fine di ottenere X$ di lunghezza uguale ad X.

Analiziamo riga per riga la routine di EXAMINE:

60100	La variabile X3$ è posta a spazio, carattere di riempimento per un input alfanumerico o numerico con zona non significativa voluta a spazi.
60110	Se X$ di partenza è uguale a N (numero), la zona non significativa dovrà essere composta da zeri e quindi il contenuto di X3$ è variato opportunamente.
60115	Se X$ di partenza è uguale a Z (numero), la zona non significativa mantiene il valore di spazio e il valore X$ è salvato in X4$.
60120	Si esegue un normale comando di INPUT.
60130	Inizializzazione del valore di LX, lunghezza di X$.
60131, 60132	Controllo di numericità.
60133, 60136	Loop di controllo di numericità del contenuto della variabile X$. Il controllo viene eseguito byte per byte.
60141	Se la lunghezza di X$ corrisponde a quella richiesta si esce dalla routine.
60150	Se la lunghezza di X$ eccede quella richiesta si esegue il troncamento.
60160, 60200	Dato che a questo punto l'argomento di X$ è minore di N, è costruita la stringa X2$ da aggiungere a X$.
60210	Se il contenuto di X$ è numerico, gli spazi o gli zeri non significativi dovranno risultare a sinistra del numero stesso; diversamente (60220, 60230) la stringa alfanumerica sarà a destra.
60235	X$ è più grande di N. Il troncamento è a sinistra per un contenuto numerico ed a destra (60240) per un contenuto alfanumerico.
60240	Uscita della routine.

La routine proposta è stata concepita per funzionare nei modi descritti. Però, se si dovessero trattare solo stringhe alfanumeriche, è chiaro che alcune istruzioni risulterebbero superflue.
In particolare è allora possibile eliminare le seguenti righe: 60110, 60115, 60131, 60132, 60133, 60134, 60135, 60136, 60210, 60235, sostituendo quindi alla riga 60150 dopo THEN, il numero 60235 con 60240.

di **Luciano Di Marco**
I0DML

Il programma di ricezione in codice morse che viene descritto è stato compilato in BASIC per il SYM-1 della Synertek e può essere usato senza modifiche anche con l'AIM65, per gli altri computer richiede solo una piccola modifica. Pur essendo scritto in BASIC è sufficientemente veloce per poter ricevere ancora perfettamente una emissione morse con una velocità di battuta di 120 c.p.m. (caratteri per minuto). Il programma ha l'adeguamento della velocità di ricezione a quella di emissione inoltre spazia anche correttamente le parole tra di loro.

Come si può notare guardando il listing esso è anche molto breve, si tenga oltretutto presente che le righe precedenti il n° 100 si possono anche tralasciare perchè servono solo ad una presentazione del titolo sul monitor.

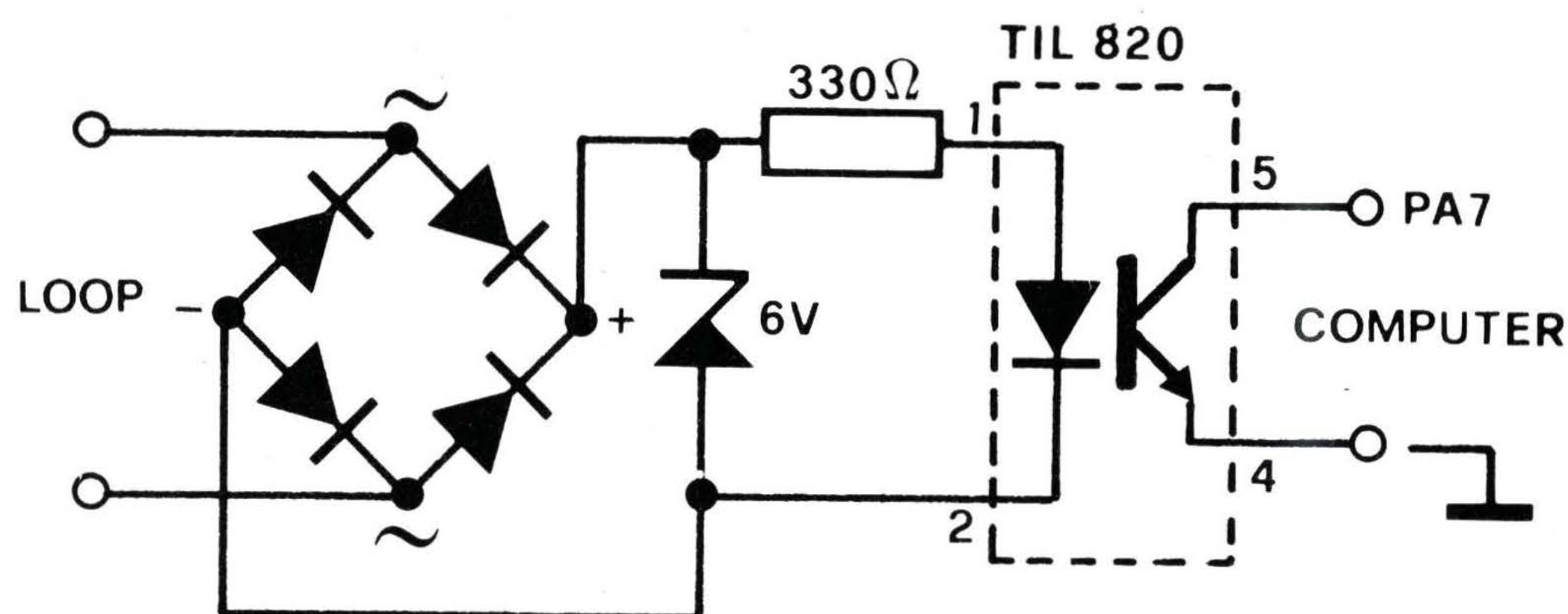
Chi dovesse adattarlo ad un altro microcomputer tenga presente che le righe da modificare sono la 130, in cui si predispone una porta, la più alta (PA7), del dispositivo I/O in posizione di input, e la riga 140 in cui si legge lo stato di tale porta (tasto alzato = 1, tasto abbassato = 0).

Il valore della velocità di ricezione dato dal valore di K nella riga 120; esso serve per distinguere i punti dalle linee. Essendo K diversamente proporzionale alla velocità, è possibile modificare questa associando a K, ad esempio K = 6, K la velocità iniziale è troppa bassa.

Descrizione del programma

Guardando lo schema di flusso riportato le considerazioni seguenti risulteranno più chiare.

Il programma essenzialmente tiene conto del tempo in cui il tasto è abbassato, incrementando la variabile N, e del tempo in cui il tasto è alzato, incrementando la variabile P. Il confronto tra queste variabili ed il valore di K determina il riconoscimento del punto, della linea, dello spazio tra segni dello stesso carattere, di caratteri differenti o di differenti parole.

All'inizio tutte le variabili valgono zero ad eccezione di K che vale 10. Se il tasto non viene abbassato la variabile P aumenta fino a diventare maggiore di K, quindi si ha la stampa di uno spazio incrementando ancora P fino a diventare maggiore di 1.5 K si ha la stampa di un'altro spazio in questo momento

RICEZIONE MORSE

```
LIST

 10 PRINTCHR$(12);:FORI=1TO100:NEXTI
 12 PRINT "    ********************************"
 14 PRINT"    *";TAB(36);"*"
 16 PRINT"    *     PROGRAMMA RICEZIONE MORSE    *"
 18 PRINT"    *";TAB(36);"*"
 20 PRINT"    *     CON ADEG. VEL. AUTOMATICO    *"
 22 PRINT"    *";TAB(36);"*"
 24 PRINT "    *     (INGRESSO PA7--CONN.A#8)     *"
 26 PRINT"    *";TAB(36);"*"
 28 PRINT"    ********************************"
 30 PRINT:PRINT:PRINT
 100 DIMA$(80)
 110 FORJ=0TO80:READA$(J):NEXTJ
 120 K=10
 130 G=40960:POKEG+3,127:POKEG+1,0
 140 Q=PEEK(G+1)
 150 IFQ<128GOTO 300
 160 IFC=2GOTO 140
 170 P=P+1
 180 IFC=1GOTO 240
 190 IFP>KGOTO 240
 200 IFN=0GOTO 140
 210 IFN<KGOTO 230
 220 K=7/8*K+N/12:A=A*2+1:N=0:P=0:GOTO140
 230 K=7/8*K+N/4:A=A*2:N=0:P=0:GOTO140
 240 IFA=0ANDP<1.5*KGOTO 140
 250 IFA=85THENPRINT".";:GOTO 290
 260 IFA=115THENPRINT",";:GOTO 290
 270 IFA>80THENPRINT"$";:GOTO 290
 280 PRINTA$(A);
 290 P=0:A=0:C=C+1:N=0:GOTO 140
 300 IFA=0THENA=1
 310 C=0:P=0:N=N+1:GOTO140
 320 DATA" ",$,E,T,I,A,N,M,S,U,R,W,D,K,G,O,H,V,F,$
 330 DATAL,$,P,J,B,X,C,Y,Z,Q,$,$,5,4,$,3,$,$,$,2,$
 340 DATA$,$,$,$,$,$,1,6,"-","/",$,$,$,$,$,7,$,$,$
 350 DATA8,$,9,0,$,$,$,$,$,$,$,$,$,$,$,$,?,$,$,$,$
 360 END
```

Programma di ricezione Morse

C = 2 e l'incremento di P si arresta.
Quando il tasto viene abbassato si incrementa la variabile N, non appena il tasto viene rilasciato la N viene confrontata con K, se è minore, il segno viene interpretato come un punto, se è maggiore come una linea. Il tempo che il tasto viene tenuto sollevato viene contato dalla variabile P che quando è maggiore di K indica che il carattere è completo, se diviene maggiore di 1,5 K indica che una parola è completa.
Il valore di K viene continuamente confrontato con i valori di N, e con peso diverso a seconda che si tratti di linea o di punto, in modo da adeguare entro 5 o 6 caratteri la velocità di ricezione.
La interpretazione vera e propria del carattere viene eseguita dalla variabile A, essa viene posto = 1 all'inizio di ciascun carattere. Per comprendere il funzionamento pensiamo esclusivamente alla sua rappresentazione in binario, su di essa operiamo uno shift a sinistra di un posto, ossia facciamo entrare uno zero da destra, se si riceve un punto, mentre diamo uno shift a sinistra e facciamo entrare un uno da destra se si riceve una linea. Alla fine del carattere avremo una sequenza di uni e di zeri che, a parte il primo uno, rappresenterà fedelmente il carattere; ad esempio la lettera "C" sarà caricata come: 11010, mentre la lettera "T" sarà: 11; quindi per la variabile A la lettera "C" sarà rappresentata dal numero decimale 26, mentre la lettera "T" dal numero 3 e così via.
Naturalmente in BASIC per avere lo shift a sinistra di un posto si deve effettuare la moltiplicazione per due, se si vuole che oltre allo shift entri un uno, basta sommare 1 dopo la moltiplicazione. Ciò è fatto nelle istruzioni delle righe 220 e 230.
Il programma nelle righe 100 e 110 carica tutti i caratteri da stampare in una variabile indicizzata A$ (J), quindi nella istruzione 280 stampa il carattere indicizzato dalla lettera A secondo le regole scritte sopra.
L'uso del programma è cosa molto semplice, ma bisogna fare attenzione a come collegare il computer con il dispositivo che converte il segnale morse dal ricevitore radio ad un contatto pulito di un relè (che simula un tasto). Infatti un tale collegamento può dar luogo ad inconvenienti causa i rimbalzi dei contatti del relè, d'altra parte è da evitare un accoppiamento diretto del computer al demodulatore morse, l'unica via che non presenta inconvenienti è l'accoppiamento con un optoelettronico del tipo mostrato in Figura 1 ■

NOTA: Per l'utilizzo del programma descritto, con PET-Commodore la variabile G di riga 130 assume il valore G = 59456

di **Roberto Bel**

In alcuni personal computers ed in molti sistemi autodidattici esiste il problema del collegamento con un monitor televisivo. Di questo problema molto sentito, soprattutto dagli sperimentatori, esistono vari approcci.
Certi sistemi prevedono una uscita a segnale video composto utile cioè al pilotaggio di un apposito monitor; altri sistemi hanno solamente tre uscite di sincronismo: verticale, orizzontale e segnale video. Quest'ultimo segnale non comprende, come potrebbe sembrare, l'insieme dei tre, ma è esclusivamente composto da un singolo segnale di comando per l'accensione o lo spegnimento del pannello elettronico del tubo catodico.
Un caso tipico è riscontrabile sul Pet 2001 della Commodore, che nonostante possegga un valido monitor, prevede le tre uscite (asincronismo orizzontale, sincronismo verticale, segnale video) sui terminali della User-Port.
L'applicazione di un secondo monitor, per un personal già dotato di video proprio, potrebbe essere rappresentata dalla necessità di uno schermo con maggiori dimensioni, oppure sfruttare il segnale presente sulla porta quale emissione utile per una TV privata. Anche per il primo caso, quello dell'uscita video e segnale composto, esistono ragioni valide per le quali si consiglia l'utilizzo di un modulatore TV.
Ad esempio evitare aspri diverbi famigliari attentando alla salute dell'elettrodomestico preferito.
Le motivazioni potrebbero essere tante altre; passiamo quindi, senza indugio, alla descrizione ed alla realizzazione pratica dell'aggeggio che autocostruiamo.
L'apparato, nel suo insieme consta di due parti ben distinte.
La prima delle quali è costituita da un semplicissimo integrato della serie TTL di reperibilità assolutamente certa.
La seconda parte, quella che comprende il modulatore TV-VHF, (v. fig. 2) è costituita da una scatola di montaggio, a basso costo, della Kuriuskit la KS340 in vendita presso un qualsiasi punto GBC. Il circuito integrato è un 74LS02 composto da quattro porte NAND a due ingressi ciascuna, solo tre di queste porte saranno utilizzate dal nostro progettino.
I tre segnali posti in ingresso vengono opportunamente miscelati utilizzando sia la logica TTL che le variazioni forzate dei componenti passivi che costituiscono il circuito.
Per la realizzazione pratica del circuitino sarà sufficiente utilizzare le basette preforate con le piazzuole di rame già predisposte.
L'uscita di questa prima parte del circuito dovrà essere connessa all'entrata del modulatore TV con un collegamento, più corto possibile, costituito da cavetto schermato.
L'uscita del modulatore, che costituisce un piccolissimo trasmettitore TV in banda I, sarà direttamente collegata alla presa di antenna del televisore.
In questo modo si realizzerà un adattamento accettabile fra il personal-computer ed il TV.
La scatola di montaggio prevede un proprio foglio descrittivo, ma sommariamente, vediamo

Utilizziamo il televisore come monitor

alcune cose per capire meglio il funzionamento.
La presenza del trimmer potenziometrico R2 consente di utilizzare il modulatore video con diversi tipi di segnali.
Regolando quindi R2 sarà possibile dosare il segnale di ingresso fino al raggiungimento di un perfetto pilotaggio del microtrasmettitore.
Un'altra regolazione consiste nella taratura del compensatore C3 che permetterà di far coincidere la frequenza di trasmissione con un canale libero da normali emissioni televisive.
Entrambi i circuiti possono essere alimentati da una normale pila, oppure direttamente dal personal-computer; il valore corretto è di 5/6 volt continui.
Il montaggio del modulatore è reso estremamente semplice grazie alla realizzazione della bobina direttamente disegnata sul circuito stampato; tutti i componenti del kit sono del tipo per montaggio verticale.
È conveniente iniziare, come sempre, dalle resistenze fisse R1-R3-R4-R5, dopo aver piegato opportunamente i relativi terminali, infilarli nei fori previsti, saldarli e tagliare con un tronchesino le eccedenze degli stessi.
Si monteranno poi i condensatori a disco C2-C4-C6-C7 e gli elettrolitici C1-C2 per i quali deve essere rispettata la polarità.
Inserendo nel circuito stampato il trimmer capacitivo C3 prestare attenzione a non deformare i reofori che sono flessibili e delicati. Procedere con la saldatura di uno spezzone di filo tra C6 e la seconda spira di L1, volendo utilizzare un canale di ricezione basso, oppure alla terza spira di L1 desiderando l'emissione su un canale più alto.
Impiegando segnali video deboli può essere opportuno modificare il valore di R1 aumentando così il valore picco-picco del segnale.
In ogni caso sarà bene non superare il limite di 150 Ω pena la scarsa qualità dell'immagine.
Effettuata la messa a punto ed ottenuto un risultato soddisfacente l'insieme potrà essere sistemato in un piccolo box metallico che evitarà, schermando i circuiti, slittamenti e disturbi; il positivo dell'alimentazione dovrà essere collegato al contenitore.
Durante l'impiego del dispositivo descritto l'antenna esterna del TV dovrà essere staccata.

■

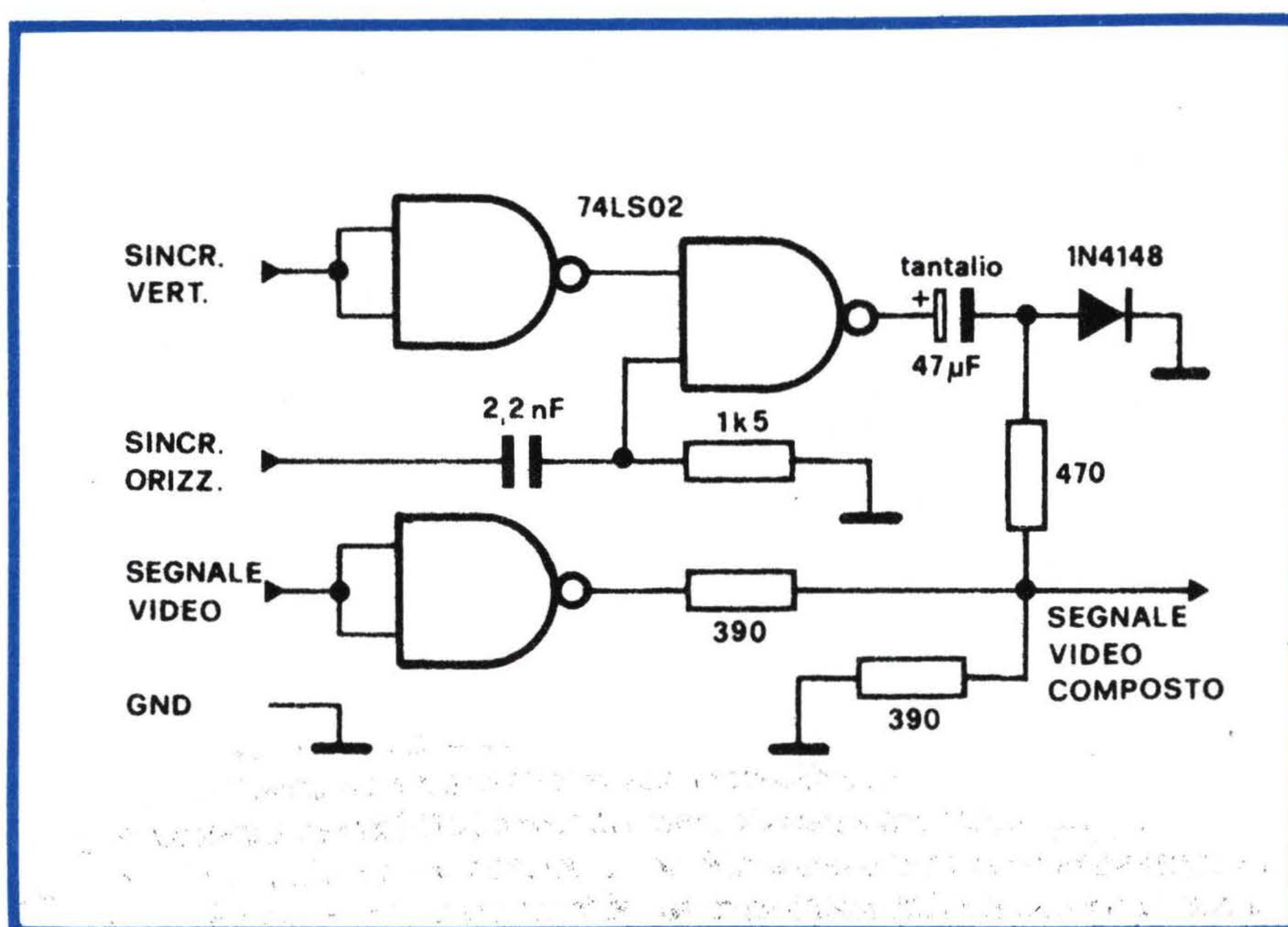



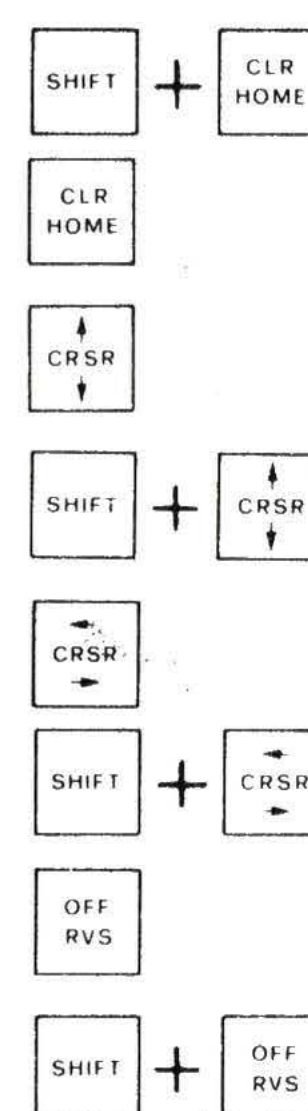
Figura 1 - Schema elettrico del miscelatore video

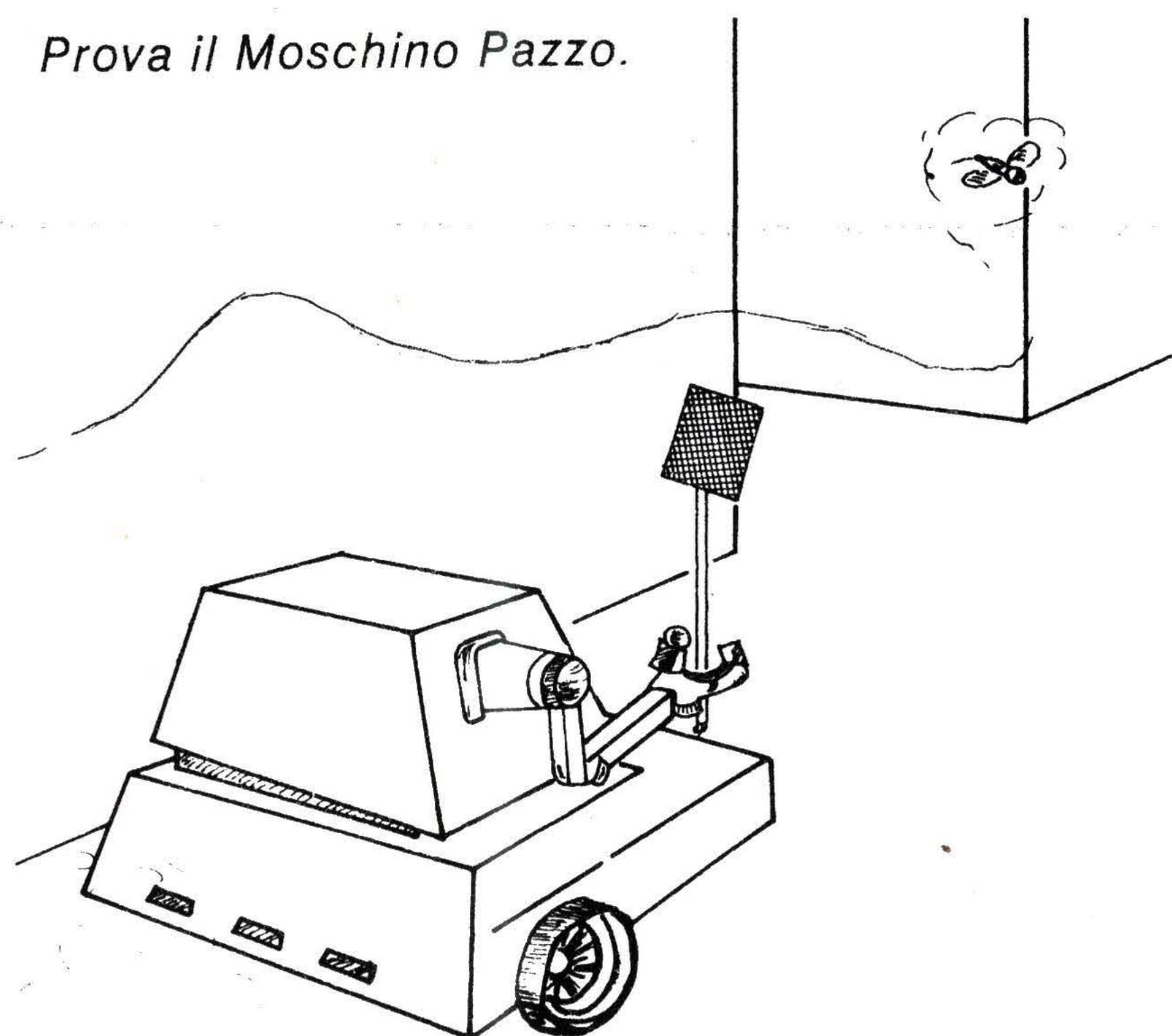
Figura 2 - Schema elettrico del modulatore TV-VHF, KS340 della Kuriuskit.

PET pratico

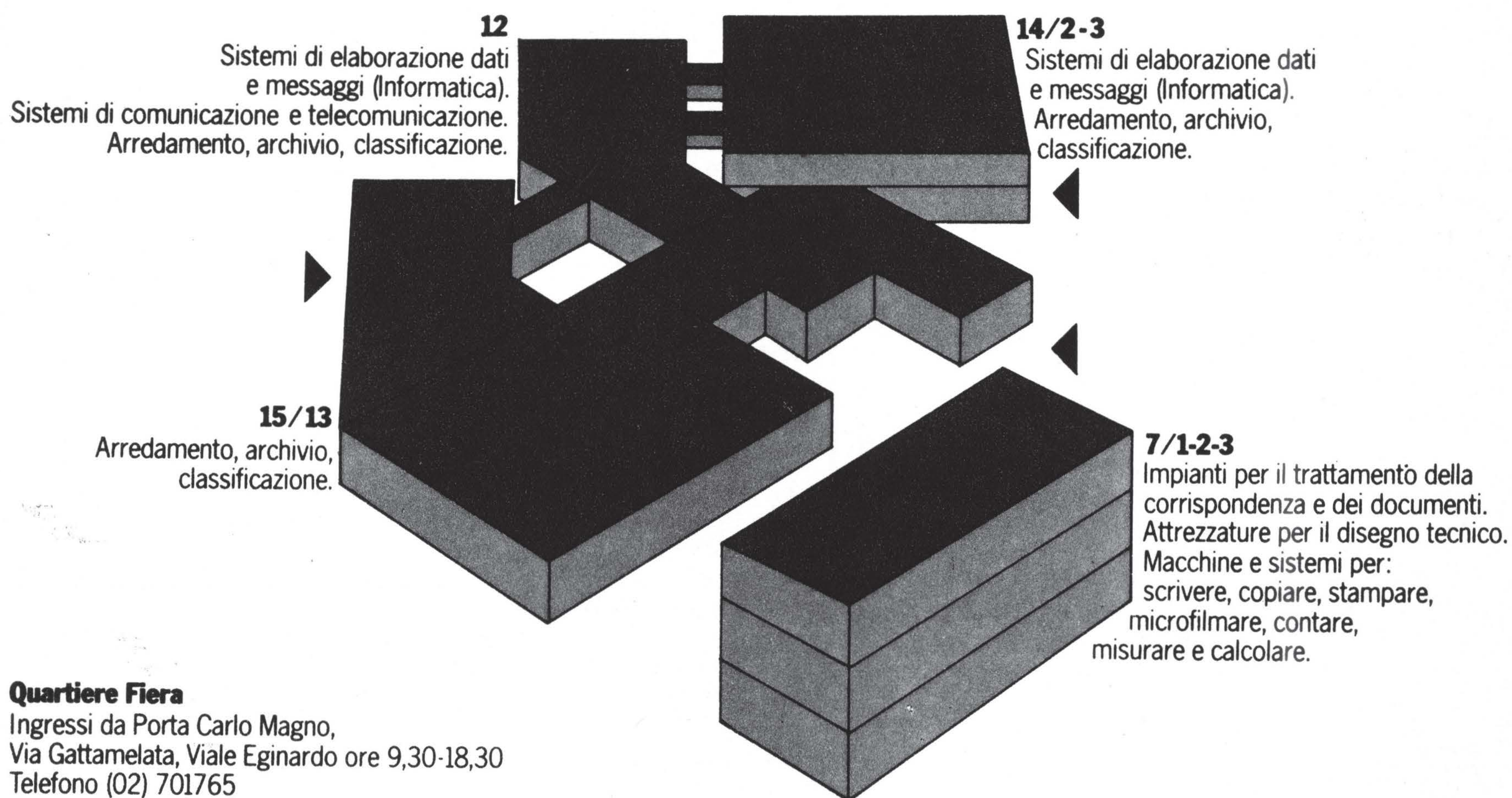
COMANDI "SPECIALI" DEL PET

La maggior parte dei programmi, scritti per il PET, contengono dei caratteri che, a prima vista, possono sembrare introvabili sulla tastiera.
Proponiamo quindi un piccolo listato per aiutare tutti coloro che da poco utilizzano questo personal.

10 PRINT" "	SCREEN CLEAR pulizia del video e cursore in alto a sinistra;	SHIFT + CLR HOME
20 PRINT" "	HOME cursore in alto a sinistra senza effettuare la pulizia dello schermo;	CLR HOME
30 PRINT" "	CURSOR DOWN cursore di una riga verso il basso;	CRSR
40 PRINT" "	CURSOR UP cursore di una riga verso l'alto;	SHIFT + CRSR
50 PRINT" "	CURSOR RIGHT cursore di uno spazio verso destra;	CRSR
60 PRINT" "	CURSOR LEFT cursore di uno spazio verso sinistra;	SHIFT + CRSR
70 PRINT" "	REVERSE quanto scritto dopo viene presentato in negativo;	OFF RVS
80 PRINT" "	OFF (REVERSE) riporta in positivo quanto viene impostato;	SHIFT + OFF RVS

```
5 PRINT"";REM                                       PULIZIA SCHERMO
10 A$=" PAGINA DEL  P E T  - ":REM                  VARIABILE DI EDIT
15 B$="":REM                                        VARIABILE BLANKING
20 PRINT"":REM                                      CENTRATURA VERTICALE
25 FORI=1TO200:NEXT:REM                             TEMPO DI ATTESA
30 A$=RIGHT$(A$,1)+LEFT$(A$,(LEN(A$)-1)):REM        ROTAZIONE DELLA VARIABILE A$
35 B$=RIGHT$(B$,1)+LEFT$(B$,1):REM                  ROTAZIONE PER BLANKING
40 A=INT(LEN(A$)/2):B=18-A:REM                      CALCOLO DI CENTRATURA SU SCHERMO
60 PRINTTAB(B)B$+A$:REM                             EDIT IN CENTRO SCHERMO
70 GOTO20
```

Prova il Moschino Pazzo.

COME ESEGUIRE:

— Centratura di una stringa alfanumerica sul video
— Rotazione della stringa
— Lampeggio della stringa
Il piccolo listato proposto serve, quale routinetta, per ottenere l'unione di particolari effetti sullo schermo del vostro Pet.
La variabile AS può contenere una qualsiasi stringa di caratteri alfanumerici (massimo 35) che sarà l'oggetto dell'elaborazione.
Nella variabile BS sono contenuti i comandi di RVS (reverse) e OFF che permettono il lampeggio.
La riga 20 comanda la centratura verticale della stringa con un'istruzione Home e tanti Cursor Down fino ad interessare la riga mediana dello schermo.
Nella riga 25 è contenuto un loop di ritardo che consente il controllo della velocità di rotazione della stringa.
Le istruzioni di riga 30 e 35 consentono l'elaborazione della stringa in modo che l'ultima cifra di AS diventi la prima e così di seguito, la medesima cosa accade per il contenuto di BS.
In riga 40 è calcolato un valore (B) che corrisponderà al comando di tabulazione dato in funzione della lunghezza del contenuto di AS, tutto questo permette la centratura orizzontale sul video.
Riga 60 Print e ritorno alla riga 20.

```
1 A$="":PRINTMID$(A$,RND(.5)*4+1,1)"";:FORI=1TO30:NEXT:PRINT"";:GOTO1
```

APPLE pratico

```
10   REM SCREEN DUMP
20   FOR I = 0 TO 2: FOR J = 0 TO 3: FOR K = 0 TO 1
30   FOR L = 0 TO 39
40   PRINT  CHR$ ( PEEK (1024 + K * 128 + I * 40 + J * 256 + L));
50   NEXT L
60   PRINT
70   NEXT K,J,I
```

La routine in alto serve per fare, su stampante, la copia esatta dell'area di memoria visualizzata sul video.
Quest'area inizia alla locazione 1024 ed arriva fino alla locazione 2047: si tratta dunque di 1024 caratteri. Dato che il video ha un formato di 24 linee di 40 caratteri ciascuna, il numero di caratteri visualizzati è 960, cioè 64 caratteri in meno della pagina di 1 kbyte dedicata al video.
Queste 64 celle sono utilizzate dai programmi di cui è fornito l'Apple II per il deposito di dati necessari al funzionamento dei programmi stessi.
Per essere più precisi questi 64 byte non sono contigui, ma dispersi in gruppi di 8 byte nella pagina di video! Non basta, anche le 24 linee di testo non sono disposte in modo contiguo (cioè così come sono visualizzate) nella pagina di memoria video, ma distribuite come appare nella tabella a lato.
Con riferimento quindi alla tabella stessa è opportuno vedere la pagina di memoria video suddivisa in blocchi di 128 byte (per un totale quindi di 8 blocchi),nel primo dei quali sono memorizzate le linee di video 1,9,17; nel secondo blocco le linee 2,10,18 e via di seguito sempre con riferimento alla figura.
Dato che in ogni blocco di 128 byte sono memorizzate 3 linee di 40 caratteri ciascuna, la differenza è data dagli 8 byte di cui abbiamo già parlato. Ciò detto è comprensibile la routine presentata che stampa l'area di memoria in modo tale da avere, sulla carta, la stessa immagine che compare sul video.

	0 – 39	40 – 79	80 – 119	127
0 riga n.	1	9	17	
128	2	10	18	
256	3	11	19	
384	4	12	20	
512	5	13	21	
640	6	14	22	
768	7	15	23	
896	8	16	24	

La seconda routine esegue il merge di due vettori ordinati, cioè inserisce i K elementi del vettore EY$ nel vettore IX$ che contiene già I elementi ordinati.
La routine è resa particolarmente semplice da una caratteristica dell'Applesoft; nella frase NEXT, che chiude un ciclo di FOR, è possibile omettere la variabile indice del ciclo (come si vede nella riga 60) ed in questo caso il NEXT si riferisce all'ultimo FOR incontrato. Inoltre se compare la variabile indice (riga 80), il NEXT chiude il FOR relativo alla variabile indicata e dealloca tutti i cicli FOR ... NEXT contenuti all'interno del ciclo chiuso con quella istruzione.

```
10  REM  MERGE DI UN VETTORE EY$ DI K ELEMENTI
20  REM  ORDINATI IN UN VETTORE IX$ DI I ELEMENTI
30  REM  ANCH'ESSI GIA' ORDINATI
40  FOR K = K TO 1 STEP  - 1
50  FOR I = I TO 0 STEP  - 1
60  IF EY$(K) < IX$(I) THEN IX$(I + K) = IX$(I): NEXT
70 IX$(I + K) = EY$(K)
80  NEXT K
90  RETURN
```

Nel nostro caso: se è vera la condizione della linea 60, viene eseguito il trasferimento IX$ (I+K) = IX$ (I) ed il NEXT si riferisce al FOR sulla variabile I.

Se la condizione non si verifica viene eseguita invece la riga 70 e alla riga 80 il NEXT K chiude il ciclo su K (quindi decrementa K e verifica di aver concluso) e contemporaneamente dealloca il FOR ... NEXT al suo interno, cioè quello relativo alla variabile I, che quindi non viene concluso perchè è già stato trovato il posto per l'elemento K[iesimo]

Mercato Computer

Nel seguire le indicazioni per "Mercato computer" provenienti dal questionario contenuto nell'ultimo numero, abbiamo voluto cogliere, oltre alla naturale richiesta di conoscere i prezzi dei vari prodotti a microprocessore, anche un'esigenza di chiarezza merceologica la più ampia possibile.

Ci siamo perciò chiesti se la pura e semplice indicazione del prezzo accanto al prodotto poteva essere sufficiente,nella schematicità di una tabella di mercato, a definire anche la corretta collocazione del prodotto stesso. Già, perché ci pare di capire che, mentre ad esempio è chiara la differenza fra una 126 e una Ritmo, nel caso dei microcomputers manca un'analoga sensibilità.

Il discorso del prezzo cioé diventa ben posto a patto di sapere che cosa si vuole acquistare.

Giunti a questo punto, ci è sembrata ben chiara l'impostazione da dare a "Mercato computer": anche in questa rubrica devono essere date indicazioni sulla collocazione dei vari prodotti.

Nella pratica questo ha voluto dire che "Mercato computer" in questo e nel prossimo numero è dedicato solo ed esclusivamente ai sistemi didattici ed hobbystici.

La lettura di questa tabella, come delle altre che seguiranno, sarà oggetto di lunghi pensieri, di conti, di domande magari imbarazzanti o magari ingenue, per cui non rimane che aggiungere alle vostre alcune considerazioni nostre.

Innanzitutto abbiamo voluto evidenziare l'approccio didattico di questi sistemi. Perché? Semplice: il mercato delle apparecchiature didattiche (e come offerta e soppratutto come domanda) ha ormai assunto una consistenza notevole. Comunque anche in questo caso occorre fare delle precisazioni: benché tutti questi sistemi possano essere utilizzati in ambito didattico, ci pare opportuno sottolineare il corredo di documentazione e di materiale didattico fornito da quei costruttori che più degli altri hanno voluto finalizzare in tal senso il loro prodotto: tipiche esemplificazioni sono l'MMD-1 e il Nanocomputer e, per un altro verso, il laboratorio portatile 5036 A della H.P.

Per quanto concerne l'approccio hobbystico, poche considerazioni: se per ottenere l'acquisto di un sistema da parte della scuola o dell'ufficio, un dipendente deve far passare sistemi di questa categoria per futuri strumenti di lavoro (ad esempio sistema di sviluppo a basso costo), faccia pure,ma non si autoconvinca di questa etichetta più del necessario. Un sistema di sviluppo decoroso costa attorno ai 20 milioni di lire: comunque, per chi è interessato ad avere un'autonomia di lavoro con poche lire (e molte limitazioni), è senz'altro opportuno guardare con estremo interesse questo tipo di prodotti e quanto il costruttore originario (o anche altri indipendenti) realizza a supporto di un'attività indirizzata verso il settore della progettazione hobbystica.

E' comunque significativo sottolineare che numerose aziende elettroniche hanno utilizzato questi prodotti per realizzare apparecchiature a microprocessore.

Un discorso leggermente diverso merita quella serie di schede chiamate "kit di valutazione", le quali, a differenza degli altri prodotti, sono realizzate dal costruttore stesso del microprocessore con l'intento di familiarizzare il futuro utente con il microprocessore in questione. Infatti, appena è annunciata una nuova famiglia di microprocessori, il costruttore spesso l'accompagna con una realizzazione minima, appunto il kit di valutazione, mediante il quale è possibile verificare le caratteristiche operative del nuovo componente.

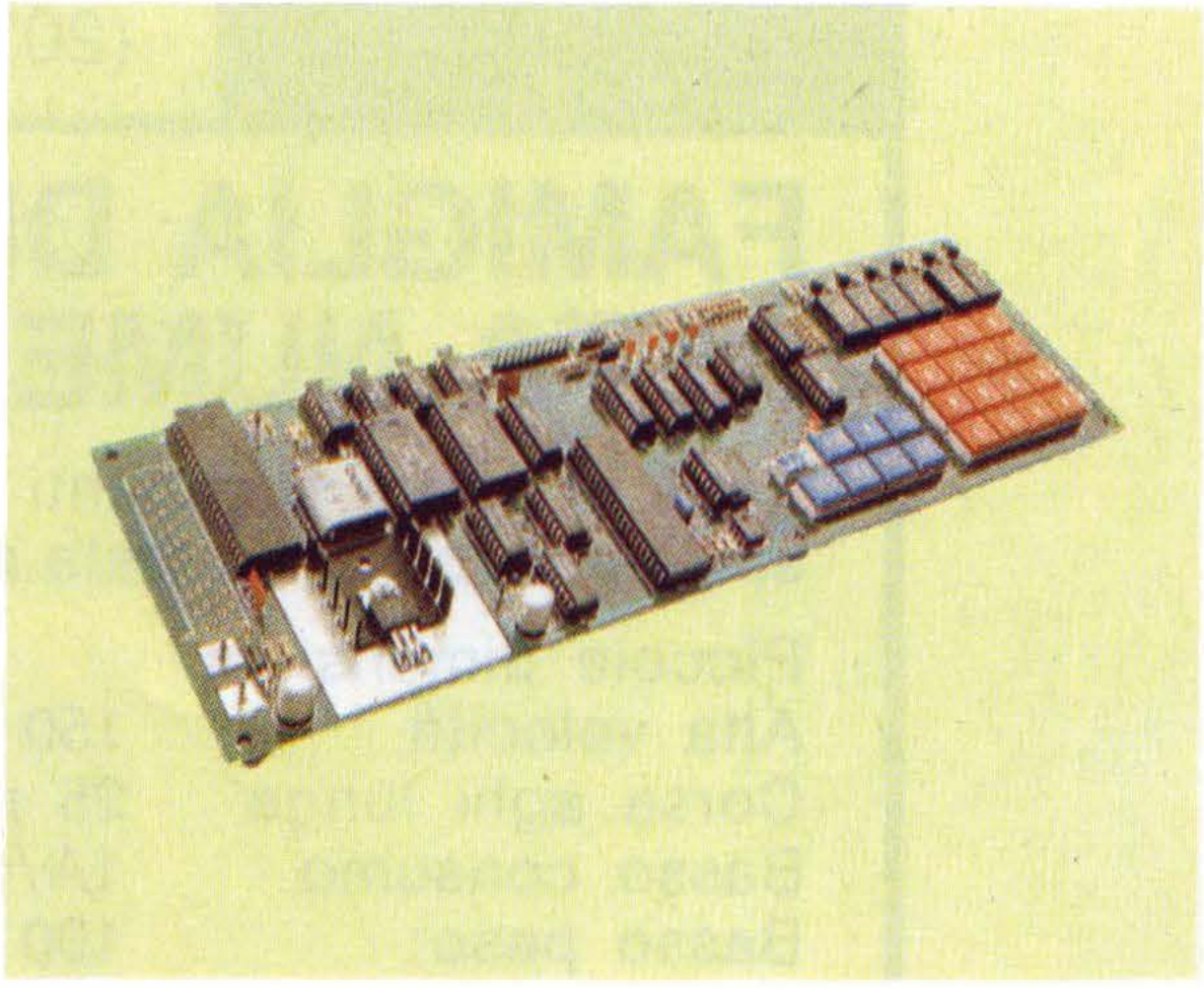
Questo approccio ha fatto sì che questo tipo di schede siano state di fatto i primi prodotti economici a disposizione degli hobbysti, ed in generale di chi è interessato a conoscere operativamente le funzionalità dei microprocessori.

Per concludere un'ultima nota: la commercializzazione di questi prodotti, date le loro caratteristiche, è fatta normalmente da strutture tradizionali come i distributori di componenti elettronici, presso i quali si trovano componenti, elementi di espansione, e quanto occorre per una crescita "normale" del sistema stesso. Per i prodotti più specificatamente didattici si hanno anche canali di distribuzione di diversa natura, appunto per venire meglio incontro al tipo di richiesta proveniente da chi è interessato ad un approccio puramente didattico.

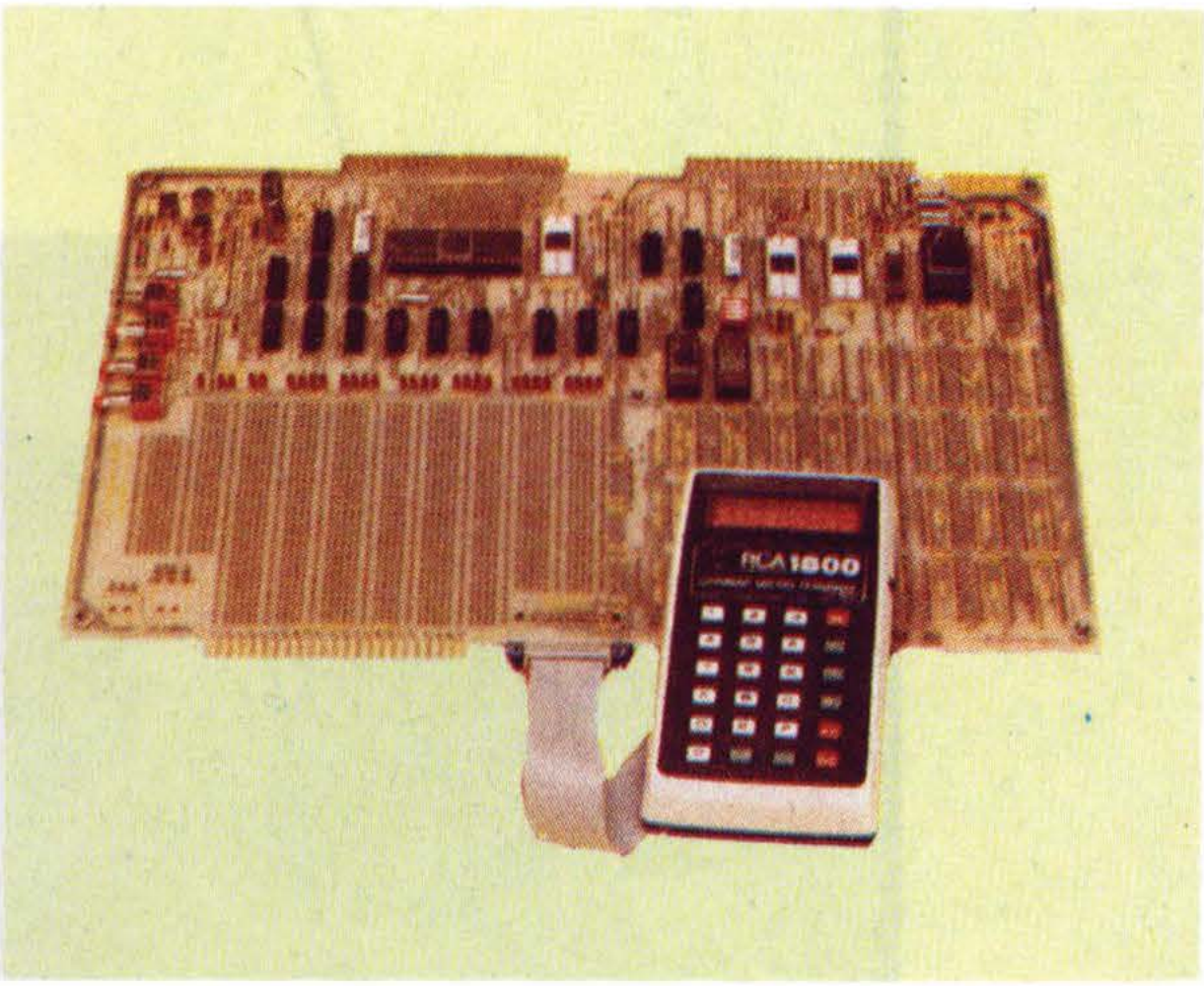
CPU	MEMORIA	INTERFACCE su scheda	ESPANSIONI su scheda	PERIFERICHE I/O	ALIMENTAZIONI	DIMENSIONI	PREZZO	MANUALI	SUPPORTI DIDATTICI
6502 8 Bit	RAM: 1 kbyte 4 kbyte ROM: 8 kbyte	Seriale RS 232C/20 mA parallela 20 linee I/O programmabili 2 registratori a cassette Stampante e TTY	RAM: 3 kbyte (per la versione da 1kbyte) ROM: 12 kbyte	Tastiera alfanumerica 54 tasti Display alfanumerico a LED 20 digit - 16 segmenti Stampante termica 20 colonne	+ 5 Vcc. + 24 Vcc.	Scheda 300 x 250 Tastiera 300 x 100	L. 535.000 + IVA (1 kbyte RAM) L. 595.000 + IVA (4 kbyte RAM)	Manuale d'uso Manuale BASIC, Hardware e Software 6502 in inglese	Giornale bimestrale "Interactive" Rokwell Int. Electronic Devices Div. Newsletter RC 55 P.O. Box 3669 Anaheim CA 92803 abbonamento $ 8,00

AIM 65
Rockwell Int. (U.S.A.)
Ing. De Mico - Milano
Via Manzoni, 51

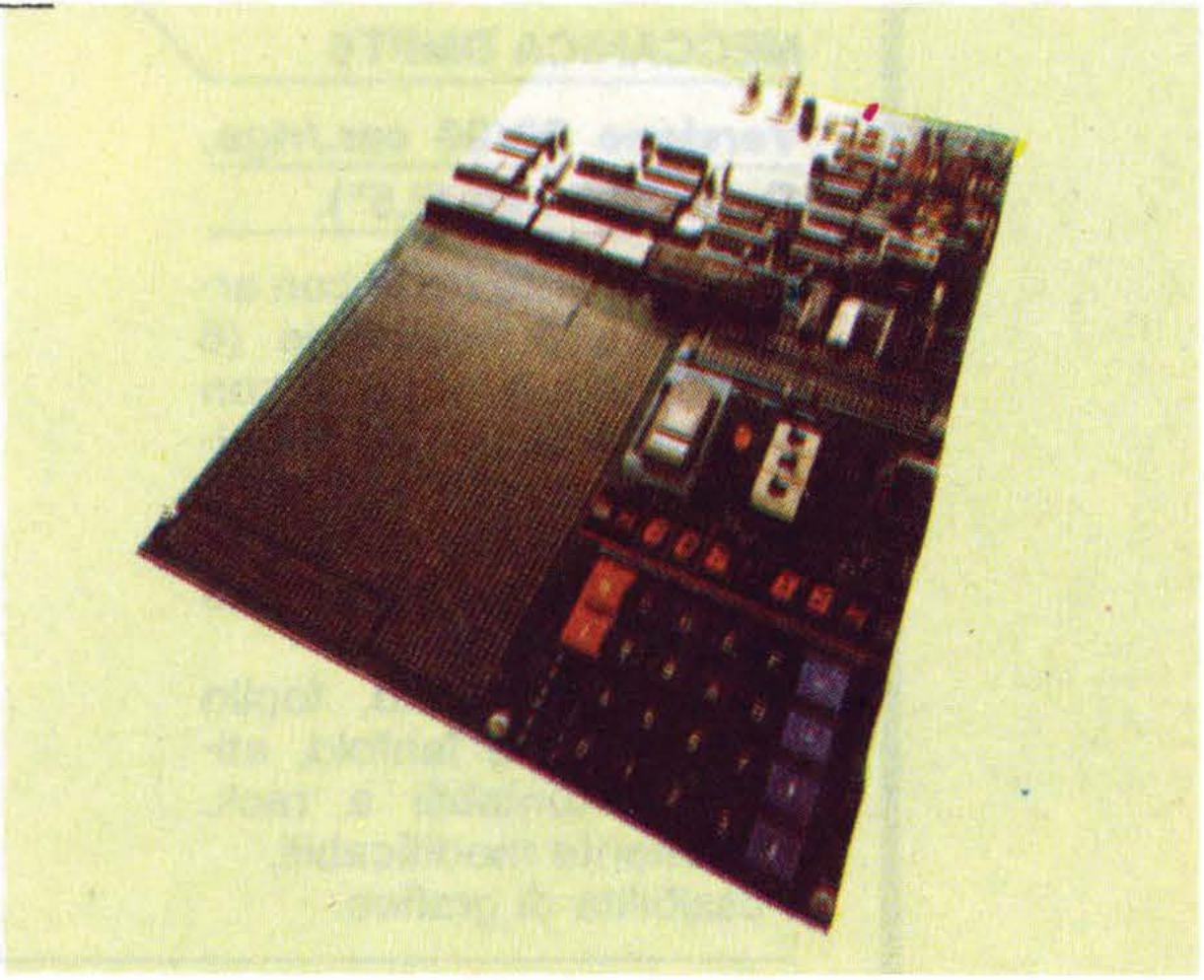
AIM 65 ESPANSIONI: BASIC 4 K ROM L. 140.000 + IVA. Assembler 4K ROM L. 119.000+ IVA. PL/65 8K ROM L. 200.000+ IVA. Programmatore di EPROM L. 95.000 + IVA. Scheda espansione memoria 16K RAM L. 545.000 + IVA. Interfaccia per monitor TV (16 righe-64 caratteri) L. 345.000+ IVA. Alimentatore +5/+24 L. 80.000+ IVA. Scheda minifloppy disk controller L. 345.000+ IVA.

CPU	MEMORIA	INTERFACCE su scheda	ESPANSIONI su scheda	PERIFERICHE I/O	ALIMENTAZIONI	DIMENSIONI	PREZZO	MANUALI	SUPPORTI DIDATTICI
6502 8 Bit	RAM: 1 kbyte ROM: 1 kbyte	1 porta I/O 8 bit (8522) 1 registratore a cassette	———	Tastiera esadecimale + 7 tasti di funzione Display a LED 6 digit 7 segmenti	+5 Vcc.	300 x 100	in kit L. 249.500 + IVA montato e collaudato L. 305.300 + IVA	———	pubblicazione "microelaboratore elettronico" edizione JCE Cinisello B. Milano

AMICO 2000 ESPANSIONI: alimentatore 1A per scheda base L. 16.500+IVA (A2000/3K). Scheda espansione per 9 schede formato Eurocard L.93.000+IVA (A2000/6). Alimentatore di potenza +5Vcc./8A - ±12Vcc./0,8A- –5Vcc./0,5A in kit L. 114.000+IVA (A2000/7K) montato L. 144.000+IVA (A2000/7). Contenitore di sistema L. 144.000+IVA (A2000/9K) completo di scheda espansioni, alimentatore di potenza L. 350.000+IVA (A2000/10). Scheda interfaccia video (16 righe-64 caratteri) in kit L. 224.000+IVA (A2000/11K) montata L. 249.500+IVA (A2000/11). Tastiera ASCII in kit L. 129.000+IVA (A2000/14K) montata L. 144.000+IVA (A2000/14). Scheda RAM/ROM con tiny BASIC da 2K ROM e 4K RAM possibilità di 16K ROM con decodifica tramite PROM in kit L. 224.000+IVA (A2000/16K) montata L. 249.000+IVA (A2000/16).

AMICO 2000
Asel Milano
Via Cortina d'Ampezzo, 17

CPU	MEMORIA	INTERFACCE su scheda	ESPANSIONI su scheda	PERIFERICHE I/O	ALIMENTAZIONI	DIMENSIONI	PREZZO	MANUALI	SUPPORTI DIDATTICI
CDP 1802 8 Bit	RAM: 256 byte ROM: 512 byte	Seriale RS 232/C 20 mA	RAM: 4 kbyte ROM: 512 byte area per montaggi wire-wrap	Terminale tipo calcolatrice tascabile 22 tasti dei quali sei di funzione Display a LED 8 digit 7 segmenti	+ 5 Vcc.	355 x 254	L. 220.000 + IVA	in inglese	———

COSMAC Evaluation kit L'evaluation kit è fornito in due configurazioni: kit CDP 18 S 020 con 256 byte memoria RAM e monitor al prezzo indicato in tabella. Kit CDP 18 S 024 con 4 kbyte RAM, Assembler Editor su banda perforata o cassetta magnetica L. 765.000+IVA. Terminale tipo calcolatrice tascabile CDP 18 A 021 L. 100.000+IVA.

COSMAC Ev.Kit
R.C.A (U.S.A)
Silverstar - Milano
Via dei Gracchi, 20

CPU	MEMORIA	INTERFACCE su scheda	ESPANSIONI su scheda	PERIFERICHE I/O	ALIMENTAZIONI	DIMENSIONI	PREZZO	MANUALI	SUPPORTI DIDATTICI
SAB 8085 8 Bit	RAM: 1 + 1/4 kbyte ROM: 2 kbyte	44 linee di I/O parallele Registratore a cassette Programmatore di EPROM	previsto uno zoccolo per EPROM (2758 o 2716) oppure ROM area per montaggi wire-wrap	Tastiera esadecimale +6 tasti di funzione Display LED 8 digit 7 segmenti	+ 5 Vcc.	230 x 320	L. 800.000 + IVA	in italiano	———

ECB 85 System bus interface per espansioni con moduli del sistema AMS 85.

ECB 85
Siemens Elettra S.p.A.
Milano - Via F. Filzi, 25/A

P | A

TRAP FALLS ROAD, SHELTON, CONNECTICUT 06484

PRACTICAL AUTOMATION, INC.

(203) 929-5381 TELEX 96-4217

FAMIGLIA DI STAMPANTI ALFA NUMERICHE

Tutte le nostre stampanti alfanumeriche si basano sull'impiego di una testa a 7 aghi di caratteristiche eccezionali:

Piccole dimensioni	
Alta velocità	150 car/sec
Corsa aghi lunga	25 mils
Basso consumo	1A/solenoide
Basso peso	100 gr
Lunga vita	100 Milioni caratteri

STAMPANTI ALFANUMERICHE PER O.E.M.

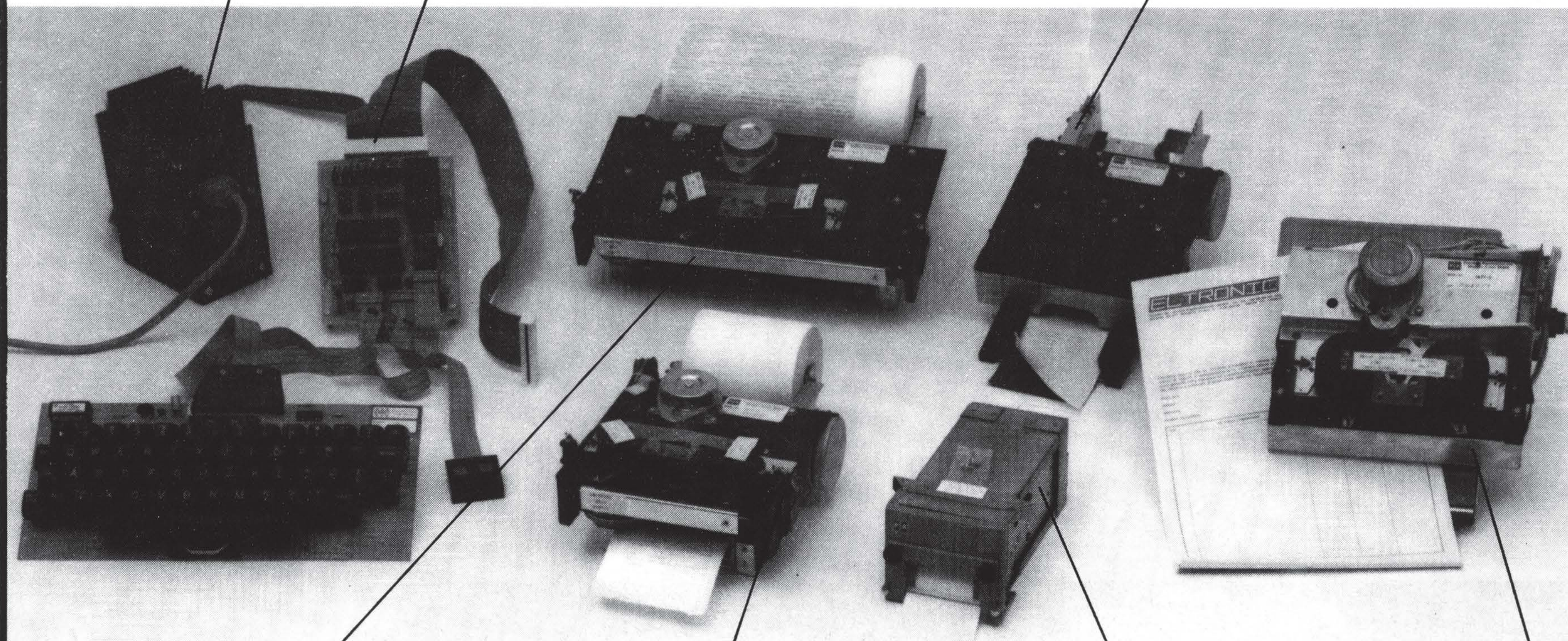
Caratteristiche comuni:
velocità di scrittura 120 car./sec. (60 Hz)
larghezza carattere variabile da 8 car./pollice a 16 car./pollice
stampa su carta comune originale e tre copie

Alimentatore universale PS 6-28 con circuito di protezione della testa ad aghi.

Interfaccia universale.

TIPO UP 72
Buffer 80 car.
Ingresso ascii parallelo.
Ingresso seriale a R5232C opp. 20 mA current loop.
Fino a 1200 Baud.
Set 64 caratteri.
Matrice 7x5 - 7x10.
Stampa in continuo fino a 300 Baud.
Riconosce LF - CR - SOH.

TIPO JX 80
Buffer 96 car.
Ingresso ascii parallelo.
Ingresso seriale RS 232 C opp. 20 mA current loop.
Fino a 9600 Baud.
Set 96 caratteri.
Matrice 7x5 - 7x10 - 14x10.
Possibilità di grafico (8192 punti/pollice quadrato).
Accetta 32 comandi software.

MECCANICA DMTP9
30—48 caratteri/riga.
Motore a passo.
Stampa su cartellini (pesatura) formato scheda meccanografica o biglietti inseriti dal davanti.
Lunghezza cartellino variabile.
Possibilità di grafico e scrittura.

MECCANICA DMPT6
Versione 80/96 car./riga.
Carta 216 mm (8,5").
Avanzamento carta con arpionismo e solenoide (6 linee/pollice) oppure con motore a passo (0,110 pollici oppure 0,166 pollici).
Stampa tipo dati (da destra a sinistra) oppure tipo testo (da sinistra a destra).
Stampa su rotolo, foglio singolo, carta fanfold, etichette. Montabili a rack.
Facilmente modificabili.
Possibilità di grafico.

MECCANICA DMPT6
Vers. 30-36-48 car./riga
Carta 87 mm (3" 7/16)
Avanzamento carta con arpionismo e solenoide (6 linee/pollice) oppure con motore a passo (0,110 pollici oppure 0,166 pollici).
Stampa tipo dati (da destra a sinistra) oppure tipo testo (da sinistra a destra).
Stampa su rotolo, foglio singolo, carta fanfold, etichette. Montabili a rack.
Facilmente modificabili.
Possibilità di grafico.

MECCANICA DMTP3
Da pannello estraibile
20 caratteri/riga.
Carta 57 mm.
Tampone inchiostrato anziché nastro.

MECCANICA DMTP 8
Da 40 a 64 caratteri/riga.
Inserzione ed estrazione carta su tre lati.
Adatta per foglio singolo, fascicoli fino a spessore 0,015", biglietti, invalidazione documenti di qualsiasi formato.

ELTRONIC ELTRONIC S.R.L.
CORSO SEMPIONE 60 - 20154 MILANO - TELEX 332695

LOMBARDIA - (02) 313364-3492615
PIEMONTE - (011) 472789
LIGURIA - (0185) 303100
VENETO - (049) 24722
EMILIA ROMAGNA - (051) 304684
TOSCANA - (055) 412018
LAZIO E CAMPANIA - (06) 6133060-6133025
SICILIA - (0942) 744691-744600

CPU	MEMORIA	INTERFACCE su scheda	ESPANSIONI su scheda	PERIFERICHE I/O	ALIMENTAZIONI	DIMENSIONI	PREZZO	MANUALI	SUPPORTI DIDATTICI
8085 8 Bit	RAM: 1 kbyte ROM: 2 kbyte	8 linee di I/O parallele programmabili BUS dati, indirizzi e controlli disponibili sull'interfaccia	———	Tastiera esadecimale +10 tasti di funzione Display a LED 6 digit, 7 segmenti 8 switches per simulazione input 8 led di simulazione output	+ 5 Vcc.	Fornito in valigetta UNI A3	L. 1.000.000 (prezzo indicativo)	in inglese	Testi italiano "Practical Microprocessor" edizione Jackson italiana Editrice Milano

HP 5036 A
Hewlett Packard - Cernusco S.N.
Via G. di Vittorio, 9

HP 5036 A Sistema previsto per l'insegnamento dell'individuazione dei guasti nei sistemi a microprocessore. La caratteristica principale dell'HP 5036 A consiste nella possibilità di simulare malfunzionamenti (corti circuiti, interruzioni di piste etc.). E' consigliato per il tipo di sperimentazione descritto, l'uso del 5004 A (Signature Analyze HP). E' possibile inoltre il collegamento con il 5024 A (Logic Troubleshooting Kit) costituito dal 545 A (Logic Probe), 546 A (Logic Pulser) ed il 547 A (Current Tracer).

CPU	MEMORIA	INTERFACCE su scheda	ESPANSIONI su scheda	PERIFERICHE I/O	ALIMENTAZIONI	DIMENSIONI	PREZZO	MANUALI	SUPPORTI DIDATTICI
6502 8 Bit	RAM: 1 kbyte ROM: 2 kbyte	Seriale RS 232C/20 mA Registratore a cassetta I/O parallelo 15 linee	———	Tastiera esadecimale +7 tasti di funzione Display a LED 6 digit 7 segmenti	+ 5 Vcc.	300 x 250	L. 250.000 + IVA	in inglese opzione manuale in italiano	Rivista "Micro" 6502 Journal P.O. BOX 6502 CHELMSFORD MA. 01824 mensile abbonamento $ 33

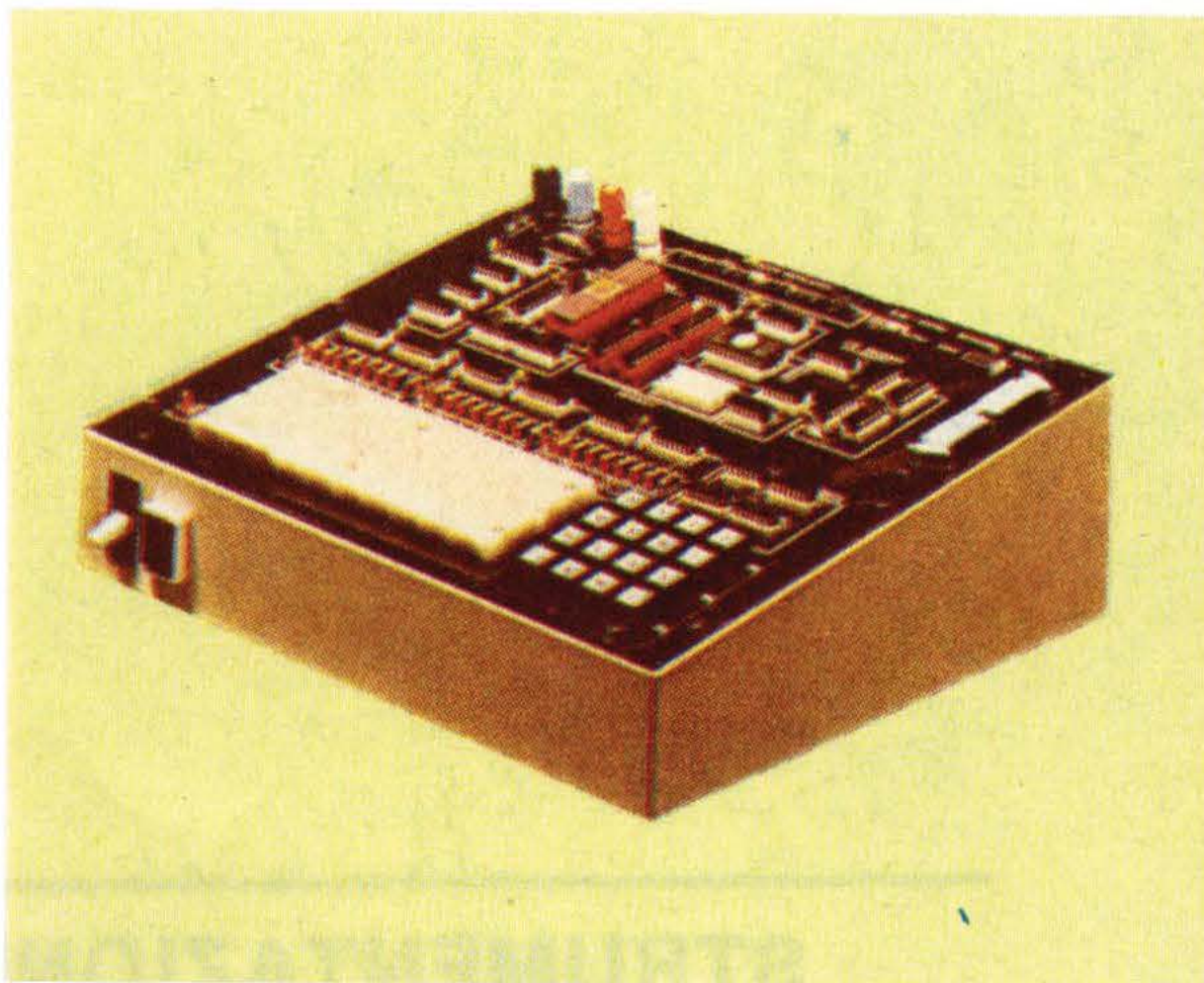
KIM 1
Mos Tecnology (U.S.A.)
Skylab - Milano
Via M. Gioia, 66

KIM 1 Il BUS Computerist permette l'espansione con le seguenti schede: scheda madre "Mother Plus" per 5 schede di espansione L. 160.000+IVA. Scheda espansione memoria "Memory Plus" 16 kbyte RAM completa di EPROM programmer e 4 zoccoli per EPROM L.390.000+IVA. Scheda espansione memoria "Dram Plus" 32 kbyte RAM completa di EPROM programmer + 4 zoccoli per EPROM + 2 VIA 6522 + area per montaggi wire - wrap L. 750.000+IVA. Scheda programmabile di gestione video "Video Plus" L.390.000+IVA. Scheda per realizzazione prototipi "Proto Plus" L. 80.000+IVA. Cestello portaschede "Cage Plus" per 6 schede L. 55.000+IVA. Stampante 25 colonne completa di software di gestione per listing e data logger L. 236.000+IVA. Scheda di gestione per floppy disk L. 700.000+IVA . Il BUS Computerist è compatibile anche con SYM 1ed AIM 65.

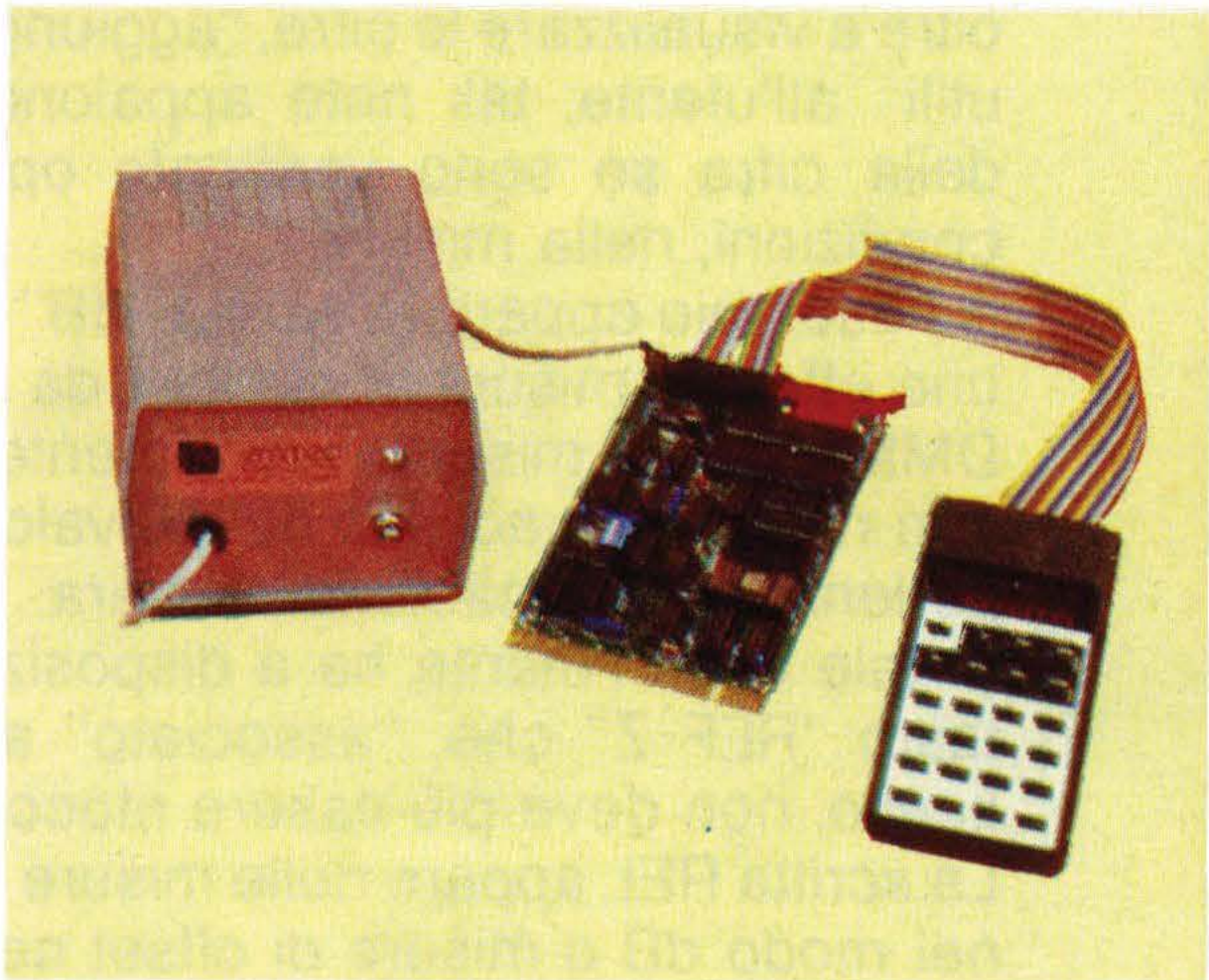
CPU	MEMORIA	INTERFACCE su scheda	ESPANSIONI su scheda	PERIFERICHE I/O	ALIMENTAZIONI	DIMENSIONI	PREZZO	MANUALI	SUPPORTI DIDATTICI
6800 8 Bit	RAM: 256 byte ROM: 1 kbyte	Parallela I/O 16 linee programmabili Registratore a cassette	RAM: 256 byte EPROM: 2 kbyte area per montaggi wire-wrap	Tastiera esadecimale +8 tasti di funzione Display a LED 6 digit 7 segmenti	+ 5 Vcc.	———	L. 347.000 + IVA	in inglese	———

MEK 6800 D2
Motorola (U.S.A.)
Cramer Italia - Roma
Via C. Colombo, 134

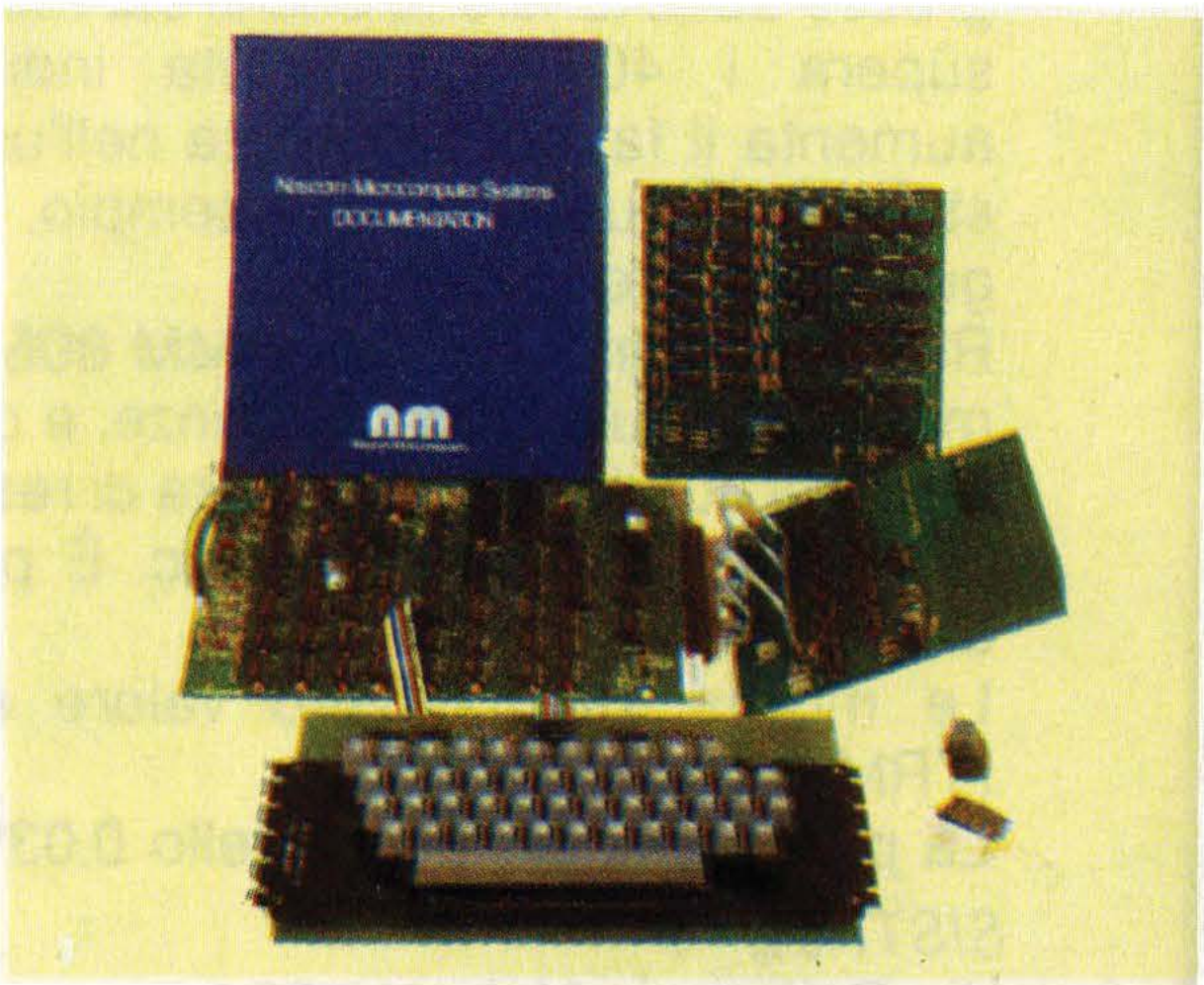
MEK 6800 D2 ESPANSIONI: scheda espansione memoria MMS 68/104 16 kbyte RAM dinamica L. 331.000+IVA. Scheda interfaccia video MEK 68 R2E L. 295.000+IVA. Tastiera ASCII in kit GRI Mod. 753 L. 98.000+IVA.

CPU	MEMORIA	INTERFACCE su scheda	ESPANSIONI su scheda	PERIFERICHE I/O	ALIMENTAZIONI	DIMENSIONI	PREZZO	MANUALI	SUPPORTI DIDATTICI
8080 8 Bit Z80 8 bit con scheda di adattamento	RAM: 512 byte ROM: 512 byte	—	Bread Board per montaggio prototipi	Tastiera esadecimale + 5 tasti di funzione Display 3 gruppi di 8 Led	+ 5 Vcc. ±12 Vcc. alimentatore compreso	254 x 305 h. 89	in kit L. 315.000 + IVA montato L. 395.000 + IVA	in italiano	Libri Bugbook Jackson Italiana Editrice Milano

MMD 1 IMMD1-Z80 scheda per utilizzo della CPU Z 80 in kit L. 136.500 +IVA, montata L. 182.300+IVA. MMD1 MI scheda interfaccia per stampante ed espansione memoria 1 kbyte RAM e zoccolo per un'altro kbyte RAM in kit L. 240.500 +IVA, montata L. 337.500+IVA. MEB 1 banco autoalimentato per espansione di memoria RAM/ROM (8 kbyte RAM/8 kbyte ROM) fornita con 2 kbyte RAM in kit L. 341.000+IVA, montata L. 444.000+IVA. VTE 1 tastiera ASCII alfanumerica con elettronica per terminale video in kit L. 651.000+IVA, montata L. 866.000+IVA. MMD 8K BASIC linguaggio BASIC da 8 kbyte fornito su EPROM L. 350.000 +IVA. OUTBOARD blocchetti premontati che consentono esperimenti di interfacciamento e conversione.

MMD1
E. & L.Instruments (U.S.A.)
Microlem - Milano
Via Monteverdi, 5

CPU	MEMORIA	INTERFACCE su scheda	ESPANSIONI su scheda	PERIFERICHE I/O	ALIMENTAZIONI	DIMENSIONI	PREZZO	MANUALI	SUPPORTI DIDATTICI
8080 8 Bit	RAM: 1 kbyte ROM: 1 kbyte	Porta I/O 8+8 linee parallele Seriale Baud-Rate programmabile	ROM: 7 kbyte	Terminale tipo calcolatrice tascabile 24 tasti di cui 7 di funzione. Display a LED 8 digit. 7 segmenti.	± 5 Vcc. ± 12 Vcc.	100 x 160	L. 350.000 + IVA Alimentatore compreso	in italiano	—

MMS 1 liv.1 Disponibili per l'espansione schede formato Eurocard. RXM-01 scheda di espansione RAM/ROM fino a 32 kbyte. RAD-01 scheda espansione RAM dinamica 32 kbyte. TVM-02 scheda interfaccia monitor TV (1 kbyte RAM). IOP-01 scheda espansione I/0 parallelo 48 linee (configurazione fissa 24 linee input TTL/24 linee output TTL). BPP-01 EPROM programmer direttamente inseribile nel BUS con moduli intercambiabili frontali per EPROM 2708, 2716 ed EPROM 2758, 2716. ARU-01 scheda per calcolo matematico (funzioni aritmetiche, trigonometriche etc.). I prezzi, in fase di revisione, non ci sono stati comunicati.

MMS-8 liv 1
L'Emmeci srl - Milano
Via Porpora, 132

CPU	MEMORIA	INTERFACCE su scheda	ESPANSIONI su scheda	PERIFERICHE I/O	ALIMENTAZIONI	DIMENSIONI	PREZZO	MANUALI	SUPPORTI DIDATTICI
Z80 8 Bit	RAM: 4 kbyte ROM: 2 kbyte	Seriale RS 232C/20 mA Registratore a cassette	RAM: 12 kbyte ROM: 6 kbyte	Terminale tipo calcolatrice tascabile 29 tasti di cui 14 di funzione. Display a LED 8 digit. 7 segmenti.	± 5 Vcc. ± 12 Vcc.	357 x 400 h. 60	L. 471.000 + IVA	in italiano	Libri NANOBOOK Jackson Italiana Editrice Milano

NANOCOMPUTER NBZ 80 NBZ 80 S come NBZ 80 B più Breadboard inserita nel contenitore L. 825.000+IVA. NBZ 80 HL come NBZ 80 S più interfaccia per monitor TV, tastiera alfanumerica BASIC da 8 kbyte su ROM più 16 kbyte RAM utente L. 1.500.000+IVA. UPZ 80 BS per l'espansione del NBZ 80 B IN NBZ 80 S L. 306.000+IVA. UPZ 80 HL per l'espansione del NBZ 80 S in NBZ 80 HL L. 750.000+IVA. KNZ 80 scheda per la conversione in CLZ 80 4/2 L.285.000+IVA. RCZ 80 registratore a cassette L. 84.000+IVA. TVZ 80 monitor TV L. 130.000+IVA. K1Z 80 fili tagliati e spelati per esperimenti L. 14.500+IVA. K2Z 80 componenti per esperimenti L.40.000+IVA. NE Z ROM per la realizzazione dei programmi del Nanobook 3 L. 45.000+IVA.

NANOCOMPUTER NBZ 80
SGS-ATES-Italia
Agrate Brianza

(continua nel prossimo numero)

STRUMENTAZIONE

MULTIMETRO FLUKE DA 4 1/2 DIGITS

La FLUKE, marca leader nel campo della strumentazione da laboratorio, annuncia un multimetro portatile e da banco, il DMM 8050A. Questo strumento presenta notevoli caratteristiche di versatilità, dovuta all'estensivo controllo delle funzioni tramite microprocessore.
Il display, a 4 1/2 digits, è a cristalli liquidi, e, oltre a visualizzare le cifre, "aggiunge" note utili all'utente; tali note appaiono a lato della cifra se sono verificate opportune condizioni, nella misura.
Ad esempio appare la scritta "dB" durante una effettiva misura in decibel: da notare il DMM 8050A misura direttamente in dB, con riferimento ad uno dei 16 valori di impedenza selezionabili da tastiera.
In tale caso l'utente ha a disposizione un tasto "REF Z" che, "associato" al valore scelto, non deve più essere ritoccato.
La scritta REL appare nelle misure relative, nel modo dB o misure di offset nelle altre funzioni.
Il tasto con la funzione associata permette di chiamare il valore di una qualsiasi lettura come "zero", e memorizzarlo: in seguito tutte le altre letture saranno riferite a quello, e sarà quindi visualizzato il valore relativo della misura; il microprocessore, che controlla le funzioni, provvede; in questo caso ad eseguire la sottrazione.
La funzione HV ricorda che il valore di ingresso della tensione che si sta misurando supera i 40 Volt: questa indicazione aumenta il fattore sicurezza nell'uso dello strumento quando, per esempio, si eseguono misure in dB.
Ricordiamo infine che il DMM 8050A permette la misura di conduttanze, e quindi la prova di condensatori, misura di resistenze di alto valore, isolamenti etc. È possibile anche la prova dei diodi.
Le misure sono il vero valore efficace (TRMS) fino a 50 kHz.
La precisione di base è dello 0.03%.

SISTREL
Via Pelizza da Volpedo 59,
20092 CINISELLO B. (MI).

MULTIMETRI DIGITALI PHILIPS A 4 CIFRE

Continua, da parte della Philips, la distribuzione di due tipi di multimetro digitale, entrambi sono dotati, per la presentazione, di un display a 4 cifre intere.
Le caratteristiche comuni sono:
- Display a 4 cifre, con indicazione del tipo di misura in atto.
- Possibilità di scelta tra un modello con display a LED ed uno con display LCD.
- Vero valore efficace nella misura delle portate in a.c.
- Elevata risoluzione e accuratezza.
- Misura di correnti fino a 10 Ampère.
- Protezione contro i sovraccarichi.

L'uso di una apposita sonda permette la misura delle temperature ed uno speciale puntale opzionale permette, durante la misura di una grandezza qualsiasi, di "congelare" l'ultimo valore assunto dalla grandezza stessa.
Il modello con il display a LED può essere posto in "standby" per migliorarne il consumo.
PM 2517E (LED)
PM 2517X (LCD)

PHILIPS S.p.A Italia.
Via Elvezia, 2
Monza

CONTATORI PHILIPS DA 120 MHz/1 GHz

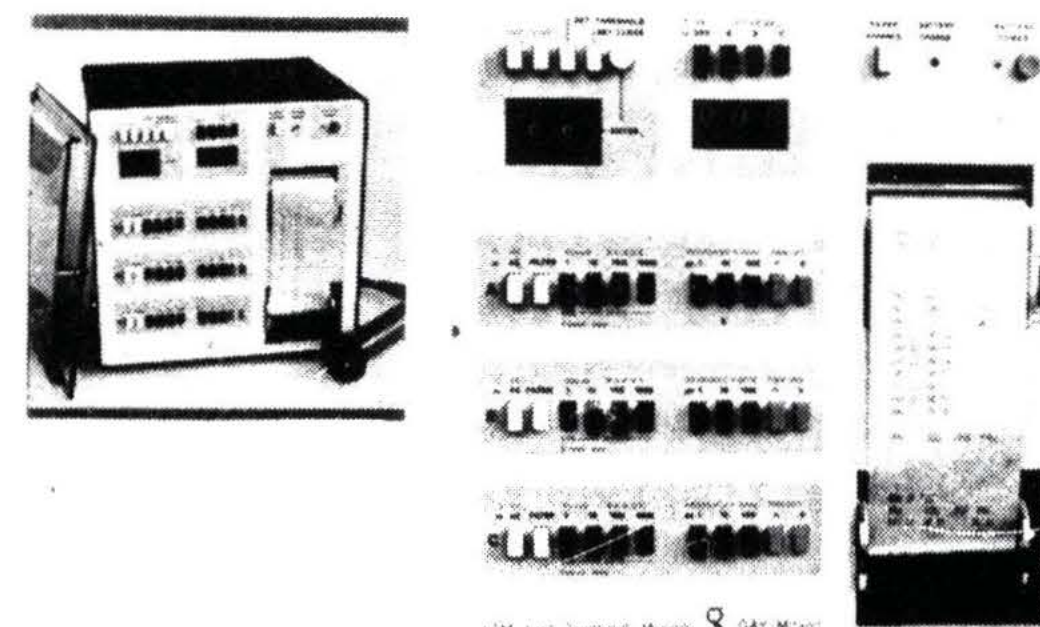
La Philips presenta due contatori ad alta risoluzione, il PM6667 ed il PM6668.
Le caratteristiche principali sono:
- controllo dello strumento a microprocessore
- conteggio reciproco di frequenza
- auto-triggering su tutte le forme d'onda
- display a cristalli liquidi
- stabilità TCXO di 10^{-7}/mese
- sottoprogramma di autodiagnosi
- dimensioni mm 160 x 77 x 180.

Il progetto, su microcalcolatore a chip singolo, consente di ottenere elevata risoluzione ed accuratezza, unitamente alla semplicità di una costruzione compatta.
L'elevata risoluzione deriva dall'impiego del conteggio di frequenza del tipo reciproco, che fornisce una risoluzione intrinsecamente elevata senza l'errore di ±/— 1 ciclo; infatti una risoluzione completa su sette cifre è ottenuta in un solo secondo. Questo procedimento evita la necessità di lunghi tempi di gate o effettuazione della misura del periodo e non presenta le limitazioni proprie dei sistemi con moltiplicatore di frequenza ad aggancio di fase.
Lo strumento "è intelligente" e fornisce automaticamente trigger e commutazione di gamma.
Esiste la possibilità di adottare gli strumenti di una base dei tempi di elevata stabilità (disponibile su opzione). Accessori disponibili sono, come sempre batteria e custodia.
L'adozione delle tecniche costruttive e di progetto descritte permettono, pur con le caratteristiche professionali indicate, un ottimo contenimento del prezzo.
PM 6668: 1 GHz.
PM 6667: 120 MHz.

PHILIPS S.p.A, Italia.
V.le Elvezia, 2
Monza

"FTM GAY" (Fast Transient Monitor)

Questo strumento è stato realizzato dalla società Gay e permette di controllare ed analizzare, in maniera continua, transitori veloci di sovratensione in linee elettriche di alimentazione e su linee di trasmissione di segnali a basso livello.
L'apparecchio "FTM" permette, in particolare, di registrare e stampare un qualsiasi evento che possa essere fonte di disfunzione in computer, terminali, strumenti a microprocessore etc.
Le prestazioni salienti dell'FTM GAY sono:
- misurazione impulsi da 4 mV a 10.000 V, durata minima di 50 nanosecondi, con una precisione del 3%;
- consumo estremamente ridotto di energia, con possibilità di funzionamento a batteria - senza interventi dell'operatore per una durata massima di 14 giorni;
- 3 canali distinti di ingresso consentono la misura su linee trifasi (o di tre funzioni su una stessa linea) grazie alla completa indipendenza di ogni canale;
- stampa automatica dei valori degli impulsi superiori a un predeterminato livello di soglia, con registrazione temporale del momento preciso in cui si è verifica-

to il fenomeno.
Nella fotografia è illustrato l'FTM con la sua stampante ed il suo quadro di comando.
L'FTM è distribuito in Italia dalla Perimel di Milano, società collegata con la Elettronucleonica di Milano.

PERIMEL
P.zza De Angeli, 7
MILANO.

COMPONENTI

DALLA A.M.D. DUE DRIVER OTTALI PER RAM MOS

La Advanced Micro Devices annuncia di aver introdotto sul mercato i driver ottali per memorie dinamiche Am 2965 e Am 2966 con uscite three-state. Entrambi i dispositivi sono stati progettati specificatamente per pilotare le caratteristiche d'ingresso capacitive delle linee di indirizzo e controllo di RAM dinamiche MOS.
I dispositivi Am 2965 e Am 2966 sono compatibili pin-to-pin con i comuni 'S240 invertente ed 'S244 non invertente. Essi offrono una migliore prestazione del sistema, limitando lo undershoot di uscita ad un massimo di -0,5V.
Caratteristica peculiare di questi dispositivi è l'inclusione di una resistenza di collettore interna nel driver di uscita inferiore. Essa controlla la caduta dell'uscita e l'undershoot, senza rallentare la salita dell'uscita. Inoltre, tale resistenza elimina lo spazio occupato da un resistore interno, indispensabile per proteggere la RAM dall'undershoot provocato dai drivers Schottky standard; d'altra parte, questi resistori esterni, rallentano la velocità di salita dell'uscita aumendando la costante di tempo RC tra driver e carico.
Per ottenere la compatibilità MOS, il driver dell'uscita superiore dell'Am 2965 e dell'Am 2966 è collegato a $V_{OH} = V_{CC}$ -1,15V.
Inoltre i driver sono progettati per ottenere un tempo di salita simmetrico con il tempo di caduta controllato dall'uscita più bassa. Ciò consente la ottimizzazione della prestazione della RAM dinamica, riducendo al minimo lo skew tra il driver superiore e quello inferiore dello stesso chip.
I dispositivi Am 2965 e Am 2966 sono caratterizzati da un tempo di propagazione (a + 2,7 V per t_{PLH} e + 0,8 V per t_{PLH}) rispettivamente di circa 10 nsec con una capacità d'ingresso di 50 pF e di 26 nsec con 500 pF. Lo skew da uscita a uscita è tipicamente minore di ± 0,5 nsec. Il normale consumo di energia di questi dispositivi a 20 pin, a soli + 5 Vdc, è di 450 mW.

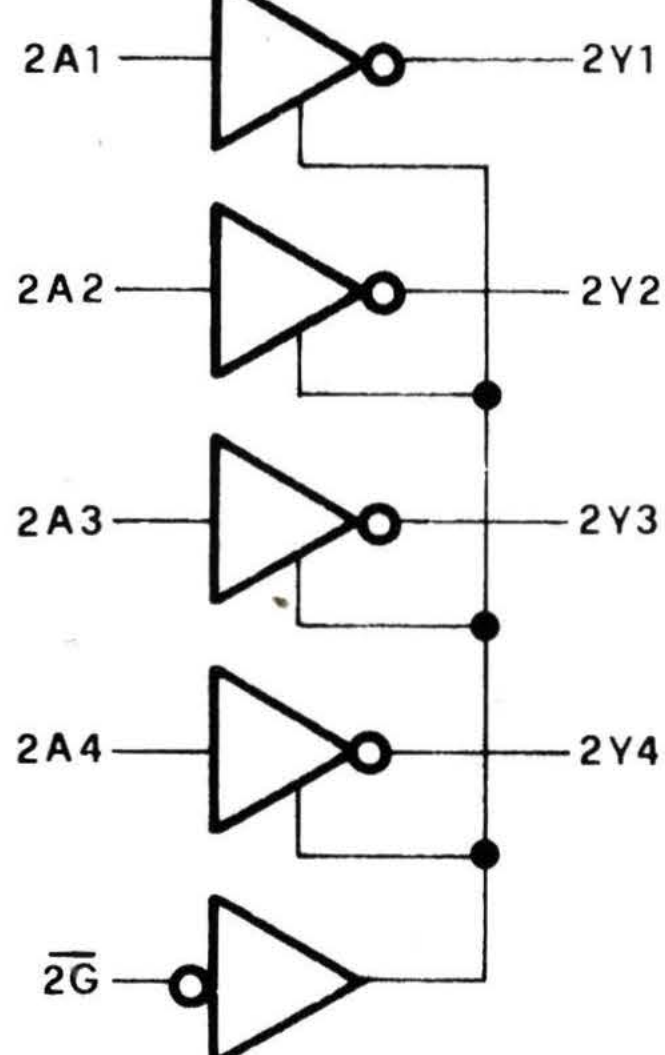
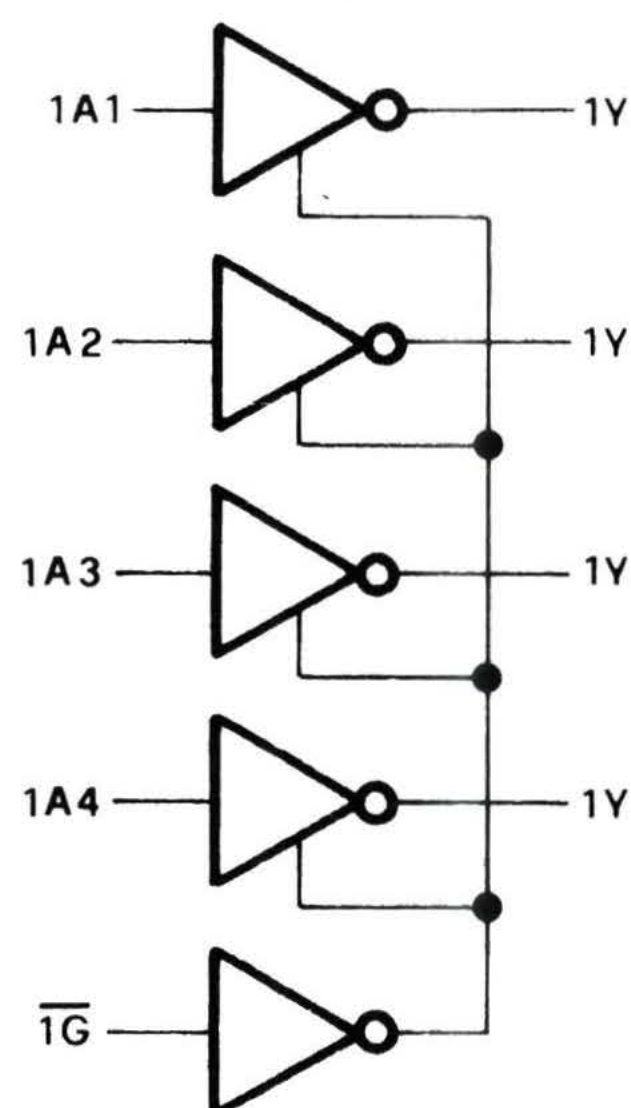
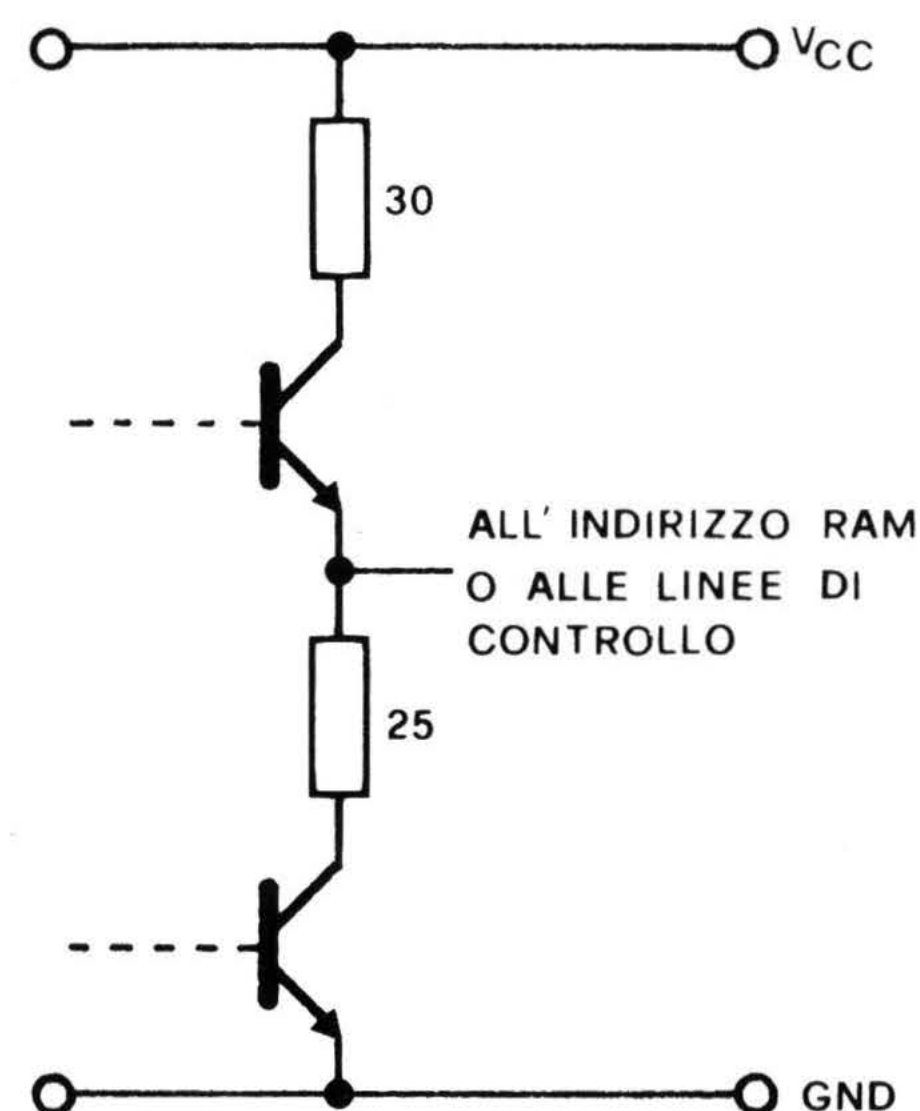
ADVANCED MICRO DEVICES srl
MILANO 2, SEGRATE

DISPLAY A BARRE DELLA LITRONIX

La Litronix ha affiancato al RBG 1000, display a barre lineari composto da 10 elementi rossi, altri due prodotti rossi e verdi ad alta efficienza denominati rispettivamente OBG 1000 e GBG 1000.
Questi ultimi display sono a dieci elementi con una lunghezza totale di 25,4 mm. Il loro contenitore a 20 pin dual-in-line permette l'accostamento di più elementi in modo da creare barre continue di differenti lunghezze e diversi colori.
Le principali caratteristiche sono catodi e anodi individuali ed una robusta costruzione.
La potenza tipica di emissione di luce, a 20 mA è stimata a 2500 μcd per l'OBG-1000 e 2000 μcd per il GBG-1000.

INTESI
Via XXV Aprile - tel. 51741.
S. Donato Milanese

DUE NUOVE FAMIGLIE TTL SCHOTTKY DALLA TEXAS INSTRUMENTS

La TI annuncia due nuove famiglie nei componenti schottky, che possono costituire una valida alternativa nei confronti della logica ECL.
Tali nuovi prodotti sono a basso consumo ed, ovviamente, ad alta velocità.
È noto che, nei circuiti logici di commutazione, la maggiore velocità si paga con la necessità di una potenza maggiore: quindi con una maggiore dissipazione per chip.
Il problema principale è dato quindi dal rapporto fra velocità e consumo, rapporto che diventa di fondamentale importanza nel campo delle logiche TTL, delle quali esiste un enorme consumo su tutti i mercati. Ora la TI propone la terza generazione dei TTL, con prodotti le cui caratteristiche si pongono ai limiti estremi dello spettro velocità-/dissipazione.
La nuova famiglia di prodotti prende il nome di "ALS", "AS"; ovvero "Advanced Lowpower Schottky" e "Advanced Schottky".
I prodotti ALS consentono prestazioni di

INPUT		OUTPUT
$\overline{G}$	A	Y
H	X	Z
L	H	L
L	L	H

velocità doppie rispetto a quelle dagli attuali prodotti LS unitamente a valori di dissipazione dimezzati.
La velocità raggiunta dai prodotti AS, doppia di quella ottenibile dagli attuali prodotti "S", allarga il campo di applicazione della logica TTL costituiscono una vera alternativa alla ECL.
Le caratteristiche generali AS sono un tempo di propagazione per Gate di 1,5 ns ed una dissipazione di 20 mW/gate.
Per le gates interne degli MSI AS si ha, tipicamente, 1 ns ed una dissipazione di 12 mW.
La nuova serie AS sarà costituita, essenzialmente, di nuove funzioni compresse a 20/24 pin, per applicazioni ad alta velocità. Verrà inoltre presentato un nuovo package da 24 pins il formato (larghezza) sarà quello tipico dei packages a 14/16 pins: ciò consentirà di raddoppiare la densità nei circuiti progettati.
La logica ALS è invece caratterizzata da 4 ns di delay e da 1 mW di dissipazione per gate.
La produzione iniziale della serie ALS sarà essenzialmente costituita dai prodotti più popolari ora in versione LS.
Le famiglie AS/ALS saranno compatibili con le attuali serie di prodotti TTL 54/74, S e LS.

TEXAS INSTRUMENTS
SEMICONDUTTORI ITALIA SPA.
Cittàducale - RIETI

MEMORIA A BOLLE MAGNETICHE DA 1 MBIT

L'Intel Magnetics 7110 è una memoria allo stato solido, non volatile, ad alta densità, da 1 megabit, che utilizza la tecnologia delle bolle magnetiche.
La capacità sfruttabile di memorizzare dati è di 1.048.576 bit. Il design insensibile ai difetti comprende storage loop ridondanti. La capacità totale della 7110 è di 1.310.720 bit.
La memoria ha una organizzazione binaria, che consente di semplificare il progetto, l'interfaccia ed il software del sistema.
Il dispositivo è organizzato in 320 loops di memorizzazione dei dati paralleli, ciascuno contenente 4096 bit.
È ammesso che 48 loops possano essere difettosi e altri 16 loop vengano riservati per il codice di correzione degli errori.
Il codice di correzione viene implementato automaticamente quando viene usato in un sistema con il formatter Intel Magnetics 7242. Quando questa memoria viene usata con la famiglia completa di elettronica di supporto della Intel Magnetics, si ottiene un sistema minimo di 128 kbytes con correzzioni automatica degli errori e manipolazione trasparente dei loop ridondanti.
La 7110 ha una architettura seriale parallela seriale con piste di lettura e scrittura separate.
L'informazione del loop ridondante è memorizzata "on chip" nel loop di bootstrap insieme al codice degli indirizzi.
In mancanza di alimentazione, la 7110 conserva i dati memorizzati fino a quando l'alimentazione stessa non sia ripristinata.
La 7110 è incapsulata in un package dual-in line completo di magneti permanenti e bobine per il campo rotante.
La memoria è protetta da uno schermo magnetico che avvolge il chip della memoria a bolle e garantisce l'integrità del contenuto contro i campi magnetici esterni.

Intel
ELEDRA 3S
Via Elvezia, 18
MILANO

PERIFERICHE

PLOTTER FORMATO A2 DALLA TEKTRONIX

La Tektronix presenta un nuovo plotter per formato A2, adatto per tutti i tipi di computer graphics, il quale può essere personalizzato dall'utente.
Il 4663 - così viene denominato il Plotter - è basato su una concezione nuova, veloce, sfrutta un notevole livello di interattività col computer ed ha una "intelligenza" propria. Offre inoltre interpolazione lineare e circolare, linee tratteggiate, generatore di caratteri standard e speciali, memoria per funzioni grafiche unitamente a molte altre caratteristiche utili.
Garantisce un'ottima qualità nel tracciato, sia che l'utente usi come supporto, carta lucido oppure mylar.
Quest'ultima possibilità d'uso rende possibile adoperare il 4663 in molti settori produttivi che si servono di supporti plastici per la grafica.
Il 4663 può inoltre usare fogli singoli o rotoli per una maggiore praticità di uso.
Infine, per chi usa il 4663 in unione al terminale grafico 4016 con schermo da 25", esiste la possibilità di un link diretto fra due moduli, in questo caso il plotter 4663, in pochi secondi, riesce a disegnare direttamente in scala uno a uno la stessa immagine che si trova sul display del terminale grafico.

TEKTRONIX ITALIANA
Via Lampedusa, 13
MILANO.

DRIVES PER CASSETTE MAGNETICHE A DOPPIA FACCIA

La MFE Corporation di SALEM, N.H., USA, annuncia un nuovo "drives di cassette magnetiche a doppia traccia". Il nuovo modello, la cui denominazione è 452B, permette la lettura e la scrittura bidirezionale su entrambi i lati della cassetta semplicemente selezionando la testina A oppure B.
In tal modo la capacità non formattata in linea è di oltre 700 kbytes.
Alternativamente la 2° traccia può venir utilizzata per la registrazione di tabelle di indirizzamento permettendo quindi l'uso del drives in un modo "random-sequenziale" diminuendo notevolmente i tempi di accesso.
Altre caratteristiche del modello 452B sono:
— testina di read after write standard
— transfer rate fino a 32.000 bps.
— tre sole parti meccaniche in movimento
— MTBF $>$ 15.000 hrs
— Error rate $<$ 1 x 10^8
— Dimensioni compatte.
L'unità si affianca ai modelli 250B, 450B, già note sul mercato (oltre 80.000 unità installate nel mondo) ed ai terminali bufferizzantti a cassetta mod. 2500 e 5000.

CONTRADATA
Via Dei Valtorta, 11
MILANO

FLOPPY DRIVES A DOPPIA FACCIA

La MFE Corporation di SALEM, N.H., USA ha iniziato le prime consegne di Floppy disck drives doppia faccia dalla fabbrica di LIVINGSTON - Scozia, ai clienti europei. Inaugurata qualche mese fa, dopo un periodo iniziale di solo assemblaggio e test, è ora pienamente operativa ed in grado di consegnare a meno di 30 giorni anche notevoli quantitativi di drives.
Le prospezioni di mercato di oltre un anno

fa, sono state largamente superate dalla realtà al punto di richiedere l'apertura di alcune filiali in Europa:

MFE - Scozia : produzione e sviluppo nuovi prodotti
MFE - Londra : commercializzazione nel mercato inglese
MFE - Germania commercializzazione nel mercato tedesco
MFE - Svizzera : commercializzazione e marketing europeo

Si va, quindi, delineando la leadership della MFE Corp. nel mercato dei Floppy doppia faccia, che vanno progressivamente sostituendo i monofaccia tradizionali in tutte le applicazioni in cui viene richiesta una maggiore capacità dati.

Le ragioni del successo MFE vanno ricercate innanzitutto nel contenuto tecnologico dei drives stessi che offrono alcune peculiarità esclusive:

— testine di tipo ceramico entrambe mobili con azionamento compensato che elimina i problemi di usura del dischetto.
— azionamento dischetto con motore a corrente continua (facilmente tamponabile a batteria) che riduce la dissipazione a soli 30W ed elimina qualsiasi jitter di lettura.
— azionamento testine a mezzo sistema Heliband (TM) con accesso di soli 3 ms. traccia/traccia.
— MTBF > 10.000 hrs.
— dimensioni che sono le più contenute tra tutti i drives presenti sul mercato.

Malgrado le caratteristiche tecniche avanzate il prezzo risulta assolutamente competitivo grazie alla locazione della fabbrica MFE nel MEC.

CONTRADATA
Via Dei Valtorta, 11
MILANO

NUOVO PERFORATORE DI BANDA

La DATA SPECIALTIES INC. di Northbrook, ILLINOIS, USA annuncia un nuovo perforatore di banda dalle caratteristiche tecniche avanzate e per alcuni versi inedite.

Una caratteristica distintiva del nuovo perforatore denominato "NC 30", è la possibilità di collegarsi ad una vasta gamma di unità di governo (CPU, Modem, CNC, NC etc.) grazie a tre differenti interfacce contemporaneamente presenti sul perforatore e semplicemente selezionabili collegando il connettore di volta in volta appropriato. L'utente con una unica unità può quindi collegarsi a scelta con: RS 232, 20 mA loop e Interfaccia parallela. Ciò è particolarmente utile nelle officine dotate di un parco macchine utensili a CN dove spesso sono presenti più tipi di interfaccia.

Caratteristiche salienti del nuovo perforatore sono:

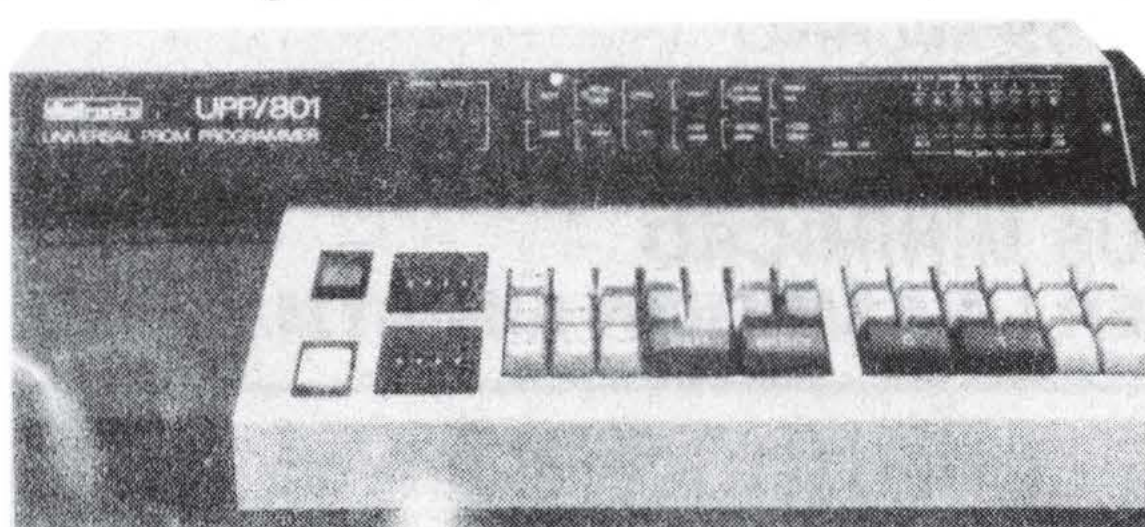
— costruzione chassis particolarmente robusta
— adozione di sigillanti e filtri per una elevata immunità alla polvere ed allo spruzzo di liquidi
— trasportabilità grazie all'apposita maniglia ed alla leggerezza (kg. 12)
— meccanica del tipo Moduperf con azionamento dei punzoni a doppio effetto e trascinamento a ruota dentata
— attitudine alla perforazione di mylar metallizzato (vita della matrice 60 Mio ch.)
— vita della matrice su carta non oleata 240 Mio ch.
— interfaccia parallela con alta immunità al rumore, con ingressi fino a 12V più opzionalmente RS 232 e TTY.
— velocità 30 cps
— garanzia 1 anno anche a clienti OEM.

Con tale unità la DATA SPECIALTIES INC. (da una recente ricerca ritenuta la N. 1 negli USA e la N. 2 nel mondo nel settore perforatori di banda) ritiene di poter ulteriormente accrescere la propria penetrazione sul mercato.

CONTRADATA
Via Dei Valtorta, 11
MILANO

UPP-801 DIGITRONICS: VERSATILE "PROM PROGRAMMER" UNIVERSALE

Il modello UPP-801 Digitronics, è un programmatore di memorie PROM di costo contenuto e ad alta versatilità.

Composto da un'unità principale di controllo e da un modulo personalizzato costituito da una scheda e da uno zoccolo d'interfaccia dedicato, l'UPP-801 permette di programmare EPROM, Bipolar Prom, PMUX, microPROM, EEPROM, EAPROM, PAL, etc, in maniera semplice e diretta in accordo alle esigenze specifiche dell'operatore.

L'unità di controllo principale è stata studiata per ospitare numerose opzioni modulari, particolarmente ideate per tenere conto sia delle necessità attuali e future dell'utilizzatore nonchè per poter lavorare come elementi d'interfaccia per la maggior parte delle periferiche di Ingresso-Uscita di normale impiego.

L'ingresso dei dati viene effettuato tramite tastiera esadecimale con diversi codici, oppure mediante computer, sistemi di sviluppo per microprocessori, telescrivente (TTY), lettori di banda perforata, con interfacciamento standard serie RS 232-C o parallelo.

L'unità di acquisizione dati RAM dell'UPP-801 può essere scelta in diversi formati di memoria con capacità di parzializzazione (splitting capability) e di accumulo (stacking capability).

Quest'ultima possibilità permette di caricare su un'unica memoria da 8 kbytes il contenuto di 8 differenti PROM da 1 kbytes ciascuna, conseguendo in tal modo consistenti risparmi di tempi operativi.

Insieme all'apparecchio sono disponibili, fra l'altro, moduli per la programmazione in parallelo di più prom, rom-Emulators, unità di cancellazione EROM a U.V.

ELETTRONUCLEONICA
P.zza de Angeli, 7
MILANO

SISTEMI DI SVILUPPO

EXORSET 30: UN COMPATTO SISTEMA DI SVILUPPO BASATO SULLA CPU MOTOROLA MC6809

La Motorola ha recentemente introdotto sul mercato il suo nuovissimo chip 6809, e ha subito creato una struttura altamente compatta ed efficiente "attorno" a questa CPU. È nato così il "computer"-"sistema di sviluppo" che prende il nome di "EXORset". Computer poichè la struttura permette all'utente "softwarista" di sfruttare a fondo le sue indubbie caratteristiche di praticità e velocità, due floppy drives incorporati ed il monitor sul quale può essere visualizzato sia il contenuto della memoria alfanumerica che il contenuto della memoria grafica. Un nutrito package software (M-BASIC, EDITOR, Assemblatore, e fra poco Pascal, Cobol etc.) e la potenza di calcolo dovuta al 6809 fanno dell'EXORset un computer veramente completo.

L'utente interessato a servirsi di un sistema di sviluppo sul quale appoggiarsi in fase di progetto col 6800/09, troverà ancora nell'EXORset un valido ausilio nel lavoro di progettazione e debug: infatti, come detto, l'EXORset è anche un validissimo sistema di sviluppo espandibile con accessori e periferiche appropriate (memorie, Programmatori di Eprom, interfaccia seriali - paralleli, e tutto quanto si adatta al System EXORciser). Un esempio per dimostrare la versatilità e la facilità nella progettazione col 6809 è dato dal fatto che una volta creato il package il software, l'utente potrà per esempio "riversarlo" su ROM/EPROM. Date le possibilità dei metodi di indirizzamento del 6809, l'utente potrà, in seguito, posizionare la sua EPROM in quella parte della mappa di memoria che preferisce, questo è consentito dalla completa rilocabilità di ogni programma così ottenuto.
La macchina si presenta in un mobile compatto, con monitor e tastiera full/ASCII.
Sono presenti due Minifloppy Disk Drives.
La memoria consta di 48 K di RAM con zoccoli per aggiuntivi 24 K di EPROM-/ROM.
Viene fornito un compilatore Extended Basic, con possibilità di funzionare come interprete.
L'utente ha la possibilità di sfruttare come preferisce i 16 tasti funzionali la tastiera ASCII "user definable". Il monitor da 9" permette grafica ad alta risoluzione, ed il video display alfanumerico consente testi di 22 linee di 80 caratteri oppure 16 linee di 40 caratteri.
Gli "slot" posti sul retro del contenitore permettono l'accesso al bus ed a tutte le linee di comunicazione fra l'EXORset ed il mondo esterno: è previsto il collegamento di una stampante standard parallela.
Naturalmente, esiste la compatibilità di bus fra EXOR set ed EXORciser.
I due drivers minifloppy permettono la memorizzazione di massa fino a 160 Kbyte ed il sistema operativo XDOS presiede il relativo controllo. Il sistema orientato verso l'utente, permette così di selezionare molte opzioni hardware direttamente da tastiera ASCII, con l'uso di apposite sequenze di escape.
Alcune di queste includono il richiamo di una mappa di memoria alternativa, di un secondo formato di visualizzazione su monitor, enable/disable graphic, etc.

CELDIS
Via F.lli Gracchi, 36
Cinisello B. - MILANO

PERSONAL COMPUTER

"SUPERBRAIN" ORA ANCHE IN ITALIA

È arrivato in Italia l'Office Computer "SUPERBRAIN", compatto con tutte le funzioni utili come scheda CPU, memoria RAM, video 12" tastiera ASCII con "Numeric Pad", n. 2 minifloppy e porte seriali, tutte incorporate in una sola custodia, di design moderno e piacevole per ogni ambiente.
I due processori Z80A lo rendono un veloce mezzo di calcolo e di ingresso/uscita con le migliori prestazioni per le applicazioni più svariate.

— La capacità di memoria interna è espandibile fino a 64mila caratteri (RAM).
— Le due unità a minifloppy in 2 versioni del SUPERBRAIN hanno una capacità complessiva di 350 Kbytes e 700 Kbytes, espandibile di un ulteriore 350 o 700 Kbytes rispettivamente.
— Le stesse unità possono essere utilizzate per il caricamento dei programmi del sistema operativo, per quello dei linguaggi, per programmi (applicativi e dei dati. Infatti sono disponibili il CP/M e i compilatori BASIC, FORTRAN, COBOL e APL.
— Inoltre il sistema base può essere potenziato integrando solo periferiche come una stampante seriele o un sistema di memoria a disco fisso fino a 100.000 milioni di caratteri rispettivamente tramite la porta seriale d'uscita principale o ausiliaria e l'adattatore BUS S-100.
— Sono anche disponibili sistemi a "multi user" e versioni potenziate con dischi rigidi fino a una capacità massima di 100 Megabytes (CDC e Sentinel) per applicazioni che richiedono maggiori prestazioni.
— Il costo dell'intero sistema base "SUPERBRAIN" nella versione standard è inferiore a Lit. 4.500.000.

MACTRONICS ITALIA,
V.le Jenner 40/A - Tel. 6882141
20159 MILANO

BUS MINIMICRO PER LA FAMIGLIA AIM/SYM/KIM

La COMPUTERJOB di Brescia annuncia l'entrata in distribuzione dell'interfaccia "MINIMICRO SISTERBOARD".
Questa scheda è nata con lo scopo di consentire, ad utenti in possesso di AIM, SYM o KIM, l'utilizzo delle schede Eurocard della famiglia "MINIMICRO", che si basano sull'omonimo bus.
Pure esistendo altri tipi di interfaccia in proposito (BIS, FLEXBUS), è stato notato che questi implicano l'adozione di una struttura a sè stante per le schede (che sono in formato Eurocard) della famiglia MINIMICRO.
Ovvero l'utente deve montare le schede Eurocard parallele una di fianco all'altra, e poi cablare il collegamento fra il BUS MINIMICRO e il bus "KIM-4" che è quello adottato, per standard, da SYM AIM e KIM nel connettore di espansione.
Con il SISTERBOARD, invece, l'utente dispone di una scheda lunga e stretta, la quale presenta da uno dei due lati lunghi i due connettori 22+22 della famiglia AIM/SYM/KIM, mentre l'altro lato lungo ospita tre femmine Eurocard a 32+32 poli che permettono al SISTERBOARD di accettare, contemporaneamente, fino a tre schede della famiglia MINIMICRO. Le tre schede montate una di fianco all'altra, danno a tutta la struttura la forma di un'unica grande scheda, su un unico piano, che ha dimensioni finali molto simili a quelle di una AIM/SYM o KIM. Il gruppo può dunque agevolmente essere montato su un piano parallelo sotto o sopra il KIM o il SYM (e sotto l'AIM), con un cablaggio, quindi, più "vicino" alle esigenze "consumer".
Le schede MINIMICRO resentano caratteristiche evolute e "concentrate" in poco spazio, così per l'utente dei suddetti microcomputer è molto facile cablare un interfaccia che già di per se porta tutti i buffer per dati ed indirizzi, oltre ad una particolare logica di controllo che permette, quando sarà operativa la scheda di controllo dei Floppy, la gestione del DMA.
La possibilità di inserire 3 schede senza nessuna ulteriori modifiche, avvantaggia l'utente che potrà, per esempio, collegare una RAM da 8K, una scheda di interfaccia parallelo/seriale (con VIA e ACIA), una scheda di memoria video a colori.
Le possibilità di espansione sono così moltiplicate per tre.

Per informazioni sul prodotto e sui distributori, contattare:

COMPUTERJOB ELETTRONICA
Via Molinari, 20
25100 Brescia.

MINICOMPUTER

SISTEMA DI GESTIONE PER PICCOLE AZIENDE

Il TA 1100 C della Triumph Adler è un minicomputer con elevate capacità di elaborazione, realizzato per dare alle unità aziendali più piccole la possibilità di disporre, a costi proporzionali, di un mini computer compatto e sofisticato.

Costituito da una unità centrale di 64 KB di memoria, un video di 1056 caratteri, una tastiera elettronica a moduli, un posto di lavoro, una o due unità floppy, una stampante ad aghi completamente isonorizzata e una vasta biblioteca programmi applicativi, il mini-computer TA 1100 C si inserisce nella gamma dei Sistemi Triumph-Adler come base per l'elaborazione distribuita, sia come computer autonomo, sia come terminale sofisticato.

TRIUMPH-ADLER ITALIA
V.le Monza, 261
MILANO

MINICALCOLATORE DA UFFICI A CASSETTE MAGNETICHE

L'unità principale del modello P 305 della Philips si presenta sotto la forma di una scrivania compatta che comprende le momorie interne, l'unità di elaborazione centrale dei dati e i diversi dispositivi di ingresso e uscita.

L'unità centrale ha una capacità di memoria di 16 kbytes, la lunghezza della parola è variabile e l'entrata del programma avviene attraverso una cassetta di ingresso-uscita del programma.

Il calcolatore è dotato di una tastiera alfanumerica con 10 tasti con 00,000 e tasto decimale; la tastiera poi ha una memoria tampone di 32 caratteri.

Il calcolatore è dotato anche di una stampante con matrice a punti, funzionante ad una velocità di stampa di 100 car/s, con una velocità di tabulazione di 300 posizioni/s, una spaziatura orizzontale di 10 car-/pollice e una spaziatura verticale di 10 righe/pollice.

Il sistema ha una velocità di trasferimento dei dati di 6000 bit/s e una densità di registrazione di 800 bit/pollice.

PHILIPS S.p.a Italia
V.le Elvezia, 2
MONZA

SISTEMA GESTIONALE DI MEDIA POTENZA

Un altro esponente della famiglia di sistemi gestionali DEC DATASYSTEM 500 basato su PDP-11/44 è stato introdotto dalla Digital Equipment.

Definito DEC DATASYSTEM 540, esso si colloca nella fascia media ma con le caratteristiche dei grandi sistemi fra cui un'insieme di istruzioni commerciali, caratteristiche per diagnostica a distanza e memoria centrale espandibile fino a 1 Mbyte di capacità.

Il DEC DATASYSTEM 540 utilizza una versione estesa del sistema operativo CTS-500 e del compilatore COBOL che offrono tempi di risposta ed esecuzione di programma del 50% più veloci di quelli normalmente disponibili su sistemi di media potenza.

Progettato per svariate applicazioni di elaborazione dati nelle banche, assicurazioni e fabbriche, il sistema può essere utilizzato come unità a se stante, oppure integrato in una rete distribuita, avvalendosi del software per comunicazioni DECnet che collega sistemi Digital fra loro ed emulatori di protocollo per interfacciare mainframes.

Il sistema è disponibile con una gamma di periferiche per la memoria di massa fra cui il sottosistema RL02, unità a disco da 10 Mbyte.

Il DEC DATASYSTEM 540 nella configurazione minima include 256 Kbytes di memoria centrale, due unità a disco RL02 (totale 20 Mbytes) e sistema operativo CTS-500. Una configurazione standard di grossa dimensione include invece 256 Kbyte memoria; unità a disco RM02 di 67 Mbytes, insieme di istruzioni commerciali, sistema operativo CTS-500 con linguaggi di programmazione COBOL-11 e DIBOL, sistemi per formattazione DECform e di inquiry DATATRIEVE.

DIGITAL EQUIPMENT
V.le Lombardia, 117
CINISELLO B.

SOFTWARE

LA ZILOG INTRODUCE UN NUOVO CROSS-SOFTWARE PER LO Z8000

A partire da Aprile la Zilog offrirà un package di software che permette agli utenti dello Z8000 di sviluppare programmi oggetto per questo avanzato micro a 16 bit lavorando sui minicalcolatori PDP-11 della Digital Equipment Corporation.

Il package di software Z8000 comprende un compilatore per il linguaggio "C", un ottimizzatore del codice oggetto generato dal compilatore C, un assemblatore Z8000, un linker ed un programma di comunicazione seriale compatibile col protocollo di trasmissione usato dalla piastra di sviluppo Z8000 DM della Zilog.

Il package lavora sotto il sistema operativo UNIX (settima versione) della Wester Electric (Bell System). L'UNIX è un sistema operativo general purpose, multi-utente ed interattivo, disponibile per i PDP-11/45 e PDP-11/70 della DEC.

Questo package si aggiunge a quello disponibile sui sistemi Z80 Zilog, che comprende assemblatore Z8000, linker e programmi di comunicazione con la piastra di sviluppo Z8000. A questo software si aggiungerà presto il compilatore PLZ/SYS nella versione Z8000.

ZELCO Via V. Monti 21
MILANO

LA ZELCO ANNUNCIA IL WORD PROCESSOR PROPORZIONALE

La Zelco S.r.l. di Milano, rappresentante per l'Italia di Zilog e Soroc, ha messo a punto una nuova versione del suo "Word Processor" in grado di trattare testi stampandoli su macchine con spaziature del carattere proporzionale, siano esse stampanti a margherita o a matrice di punti.

La versione attuale è predisposta per la nuova stampante Centronics 753, ma la struttura del programma consente di adattarlo facilmente a macchine aventi proporzioni dei caratteri e parole di controllo di tipo diverso.

Il "WP" della Zelco permette di definire le seguenti caratteristiche del testo:

— posizione dei margini destro e sinistro (assoluta e relativa)
— margine superiore
— margine inferiore
— lunghezza della pagina
— interlinea
— espansione del carattere (su stampanti Centronics)
— posizione degli "a capo"
— posizione dei salti di riga

— posizione dei salti di pagina
— capoversi di vario tipo ("indentation")
— titoli (che vengono centrati automaticamente)
— richiamo di altri file mediante comandi contenuti nel testo

È prevista l'aggiunta della capacità di numerazione automatica delle righe e la ripetizione automatica dei titoli per capitoli o paragrafi.
La flessibilità e la facilità di uso del "WP" ne fanno lo strumento ideale per la messa a punto e la stampa di testi, eliminando i tempi morti e gli errori dovuti alle correzioni e successive ribattiture dl testo, come avviene nei casi normali.
Inoltre il "WP" costituisce il complemento ideale dei sistemi di elaborazione Zilog, aumentandone la versatilità, l'utilità e la potenza.

ZELCO
Via V. Monti, 21
MILANO

PACKAGE APPLICATIVI

PROCEDURE GESTIONALI APPLE

La Softec s.r.l. presenta, per gli utenti Apple, un set di package applicativi dei quali forniamo le principali caratteristiche:

Contabilità generale

Gestisce in modo completo ed analitico il piano dei conti e le relative movimentazioni.
Il record del singolo conto comprende i seguenti campi:
codice del conto
nome del conto
bilancio
La contabilità generale permette cinque funzioni fondamentali:
prima nota contabile
lista del piano dei conti
lista delle movimentazioni
lista dei bilanci
caricamento situazione singolo conto
Le registrazioni di prima nota intervengono sul conto interessato e su quello di contropartita ed a loro volta sono memorizzate in modo analitico.
La lista del piano dei conti presenta o su un video o su carta la situazione dei conti e per ciascuno di essi le movimentazioni.
La lista delle movimentazioni fornisce, sdoppiate per i due conti relativi, tutte le informazioni sulle transazioni effettuate; lo stampato su "bollato" è utilizzabile come giornale fiscale di contabilità generale.
La lista dei bilanci presenta su video o su carta, in modo sintetico, la situazione dei conti e permette, se si è codificato opportunamente (conti finanziari, conti economici), di ottenere una situazione di "profitti e perdite".
Il programma funziona su un Apple II con 48K, 1 a 2 dischi, 1 monitor, una stampante.
Vengono gestiti fino a 1000 conti e 1700 transazioni senza saturazione dei dischi. La capacità di gestione si espande eseguendo, se sono disponibili due dischi, delle riprese di saldo.

Contabilità clienti

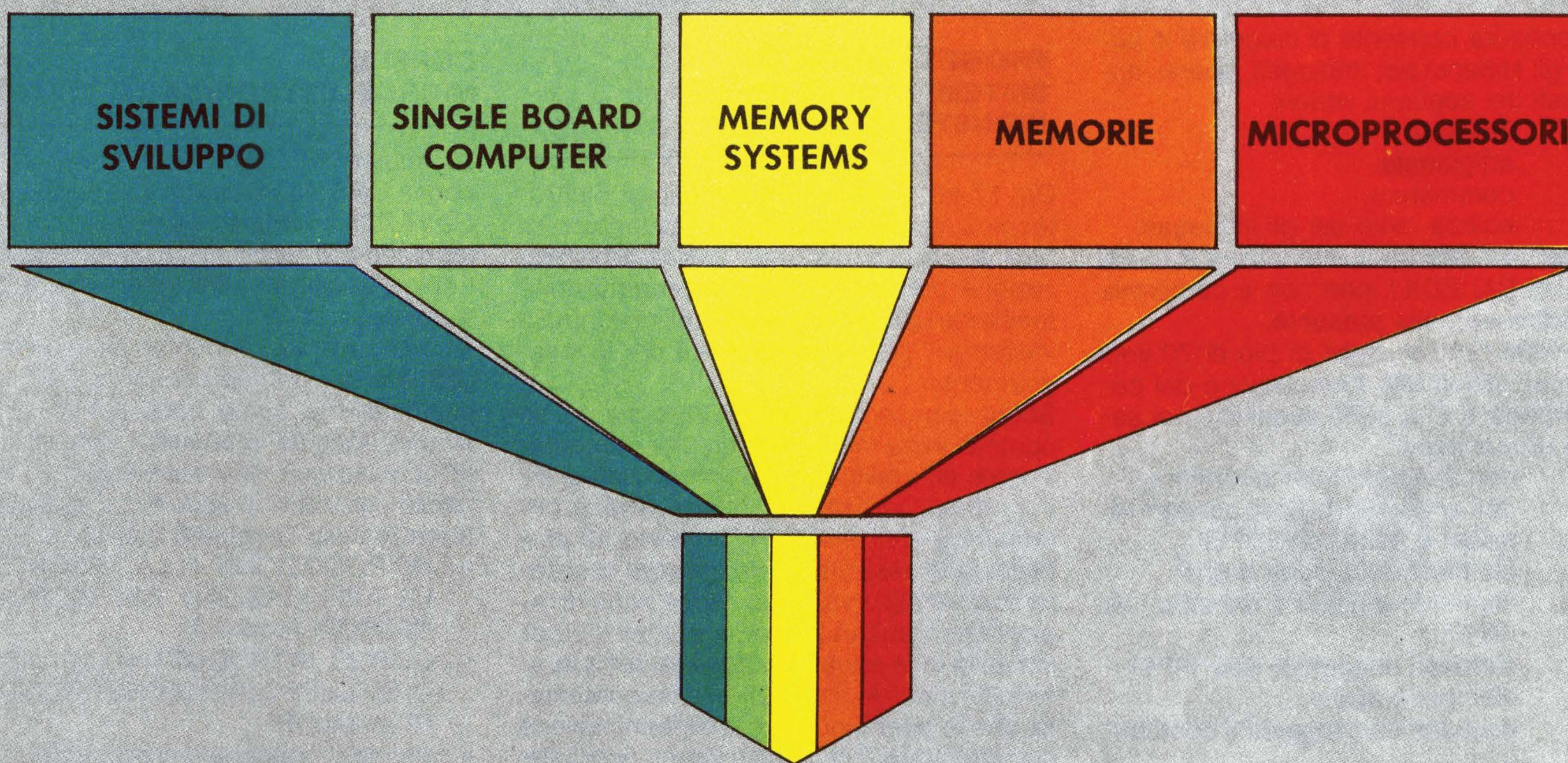
Gestisce in modo completo ed analitico i conti dei clienti e le relative movimentazioni. Il record comprende i seguenti campi:
codice cliente
anagrafica
codice fiscale
limite di credito
bilancio
progressivo imponibile
La contabilità clienti permette quattro tipi di registrazione
1 fatture clienti
2 note di credito
3 pagamenti da clienti
4 pagamenti a clienti
Le registrazioni, immediatamente imputate al conto del cliente sono memorizzate per intero ed analiticamente sul dischetto, in modo da permettere una gestione del tipo "a partite aperte".
I giornali che possono essere stampati sono quattro e relativi alle funzioni sopra citate. Gli stampati delle funzioni 1-2 se "bollati" sono utilizzati come giornali fiscali I.V.A.
Un quinto giornale è ottenibile con l'elenco di tutti i clienti che ad inizio contabilità presentavano un bilancio di apertura.
Il programma genera inoltre gli estratti conto dei clienti con tutte le movimentazioni analitiche ad essi relative.
Al richiamo su video del conto cliente è presentato in modo "inverse-lampeggiante" il bilancio nel caso sia superato il limite di credito concesso al cliente.
Il programma funziona su un Apple II con 48K, 1 o 2 dischi, 1 monitor, una stampante.
Vengono gestiti fino a 500 clienti, e 1000 movimentazioni senza saturazione dei dischi. La capacità di gestione si espande eseguendo, se sono disponibili due dischi, delle riprese di saldo.

Contabilità fornitori

Gestisce in modo completo ed analitico i conti fornitori e le relative movimentazioni.
Il record comprende i seguenti campi:
codice fornitore
anagrafica
codice fiscale
limite di credito
bilancio
progressivo imponibile
La contabilità fornitori permette quattro tipi di registrazione
1 fatture fornitori
2 note di credito
3 pagamenti da fornitori
4 pagamenti a fornitori
Le registrazioni, immediatamente imputate al conto del fornitore, sono memorizzate per intero ed analiticamente sul dischetto, in modo da permettere una gestione del tipo "a partite aperte".
I giornali che possono essere stampati sono quattro e relativi alle funzioni sopra citate. Gli stampati delle funzioni 1-2 se "bollati" sono utilizzati come giornali fiscali I.V.A.
Un quinto giornale è ottenibile con l'elenco di tutti i fornitori che ad inizio contabilità presentavano un bilancio di apertura.
Il programma genera inoltre gli estratti conto dei fornitori con tutte le movimentazioni analitiche ad essi relative.
Al richiamo su video del conto fornitore è presentato in modo "inverse-lampeggiante" il bilancio nel caso sia superato il limite di credito concesso al fornitore.
Il programma funziona su un Apple II con 48K, 1 o 2 dischi, 1 monitor, una stampante.
Vengono gestiti fino a 500 fornitori e 1000 movimentazioni senza saturazione dei dischi. La capacità di gestione si espande eseguendo, se sono disponibili due dischi, delle riprese di saldo.

Bollettazione e fatturazione

La procedura consente di emettere i seguenti documenti:
a) Bolla accompagnamento di beni viaggianti
b) Fattura accompagnatoria di beni viaggianti
c) Fattura commerciale.
La bolla (a) e la fattura (c) possono essere stampate simultaneamente, se si dispone di 2 stampanti collegate all'APPLE con 48K, monitor e da 1 a 4 dischi.
Contemporaneamento alla stampa della bolla e fattura viene aggiornato il magazzi-

no e creato un archivio I.V.A. elaborato successivamente dalla Contabilità Clienti per produrre il registro I.V.A.
Con la configurazione APPLE II dotata di 4 dischetti si hanno in linea 500 clienti e 2700 parti di magazzino al massimo. È importante ricordare che le parti di magazzino non sono limitate a 2700, ma possono essere in numero illimitato e residenti su dischetti che bisognerà inserire in macchina su richiesta della procedura.
Per gestire in modo integrato la bollettazione e fatturazione con il magazzino e con l'I.V.A. clienti, è necessario disporre di tutte e tre le seguenti procedure:
Gestione di Magazzino
Bollettazione e Fatturazione
Contabilità Clienti

Paghe e contributi

La procedura consente di calcolare le paghe e gli stipendi per impiegati, operai, apprendisti dei seguenti settori:
industria
artigianato
commercio
edilizia (solo per gli impiegati)
È disponibile anche la procedura PAGHE PER OPERAI EDILI, che non è compresa nella fornitura della presente.
Si prevede l'introduzione di più di 70 voci che concorrono alla formulazione del cedolino paga e che sono suddivise nei seguenti gruppi logici:
dati anagrafici del lavoratore;
dati anagrafici ad uso assegni familiari e detrazioni IRPEF;
elementi della paga oraria;
elementi soggetti a contributi ed IRPEF;
elementi soggetti a sola IRPEF;
elementi passivi;
elementi non soggetti a contributi;
elemento correttivo per IRPEF.
La procedura archivia automaticamente su dischetto tutti i dati, di ogni paga, che concorrono alla formazione dell'archivio storico da consultare una volta all'anno per estrarne i dati occorrenti alla compilazione dei documenti richiesti dei vari Istituti previdenziali del tipo:
modulo 101
modulo 01/M
modulo 770
modulo 03/M
modulo INAIL
Ogni dischetto può contenere l'archivio storico relativo a 70 dipendenti.
È disponibile, nella procedura, un programma che permette di pre-stampare i cedolini paga partendo da moduli continui bianchi, eliminando in tal modo le pesanti spese tipografiche richieste per la preparazione dei cedolini stessi.
La procedura funziona su un APPLE II con 48K, 1 o 2 unità a dischetti, monitor, stampante a 80 colonne di tipo seriale o parallelo.

SOFTEC s.r.l.
Via Govone, 56 - MILANO
Corso D'Azeglio, 60, TORINO

CORSI

PROGETTAZIONE DI SISTEMI DI CONTROLLO INDUSTRIALI

Dal 17 al 21 novembre l'Istituto di Elettrotecnica ed Elettronica del Politecnico di Milano terrà un corso sui "metodi di ottimizzazione parametrica per la progettazione mediante calcolatore di sistemi di controllo industriali" per un totale di 48 ore di insegnamento.
Scopo del corso è di illustrare sia da un punto di vista metodologico che computazionale ed applicativo, la potenzialità di alcuni metodi, basati sull'ottimizzazione parametrica, nell'affrontare problemi di progettazione assistita da calcolatore di sistemi di controllo industriali. Molte sono le ragioni che inducono ad utilizzare tali metodi, sia nella fase riguardante la cosidetta taratura del modello del processo da controllare, che in quella inerente la determinazione del regolatore. Tra di esse vanno ricordate: 1) la possibilità di incorporare molte delle peculiarità di un problema di progetto di un sistema di controllo, tra cui l'incertezza, la molteplicità degli scopi, i vincoli di tipo strutturale; 2) la capacità di soddisfare a requisiti di decentralizzazione degli apparati di misura, di elaborazione e controllo che sono imposti da ovvie ragioni e che rendono problematica l'applicazione dei risultati della teoria del controllo; 3) la richiesta di un impegno computazionale di tipo fuori linea e ripetitivo.
Il corso intende mettere in evidenza tali caratteristiche con riferimento a: 1) metodo di sintesi di regolatori multivariabili capaci di risolvere un considerevole numero di problemi di controllo; 2) metodi di riduzione dell'ordine di un modello complesso in presenza o no di incertezza; 3) metodi di identificazione per la taratura dei parametri del modello del processo, con particolare riferimento ai modelli ingresso-uscita (ARMAX, ecc.) e ai metodi di stima computazionalmente più semplici, come quelli a minimi quadrati; 4) metodi di controllo adattivo, per aggiornare in tempo reale i parametri del controllore a seguito di variazioni di parametri del modello del processo. Per ciascun argomento verrà presentato l'inquadramento generale, la descrizione di metodi e programmi di calcolo per risolvere i relativi problemi, l'applicazione a casi reali.

Politecnico di Milano

CORSI DI MICROELETTRONICA TECNETRA

La società TECNETRA, in collaborazione con la SGS-ATES e con la società CARLO GAVAZZI organizza una serie di corsi sulle tecnologie e applicazioni della microelettronica basati sul microprocessore Z80 e sui sistemi ZDS.
Questi corsi sono orientati all'aspetto progettuale del microprocessore Z80, quindi piuttosto che come corsi preliminari illustrativi, sono da intendersi come strumenti di acquisizione di metodologie ed esperienze utilizzabili in specifiche applicazioni.
Sono previsti i seguenti due tipi di corsi:
— INTRODUZIONE ALLA TEORIA E ALLA APPLICAZIONE DEI MICROPROCESSORI (corso A)
— CORSO INTRODUTTIVO AI SISTEMI DI SVILUPPO PER MICROPROCESSORI (corso B)
Il calendario autunnale prevede dei corsi A nelle settimane dal 22 al 26 settembre e dal 24 al 28 novembre e dei corsi B nelle settimane dal 27 al 31 ottobre e dal 15 al 19 dicembre.
Le quote di partecipazione variano da L. 400.000 (+ IVA 14%) per il corso A a L. 490.000 (+ IVA 14%) per il corso B.
Durante i corsi tenuti a MIlano presso la sede della società Carlo Gavazzi, via Ciardi 9, vengono utilizzati, per le esperienze pratiche, i sistemi didattici NBZ80 ed i sistemi di sviluppo ZDA.
Per l'iscrizione ed informazioni rivolgersi alla dr.ssa Claudia Comolo, Piazzale Morandi n. 2, Milano.

Al di sopra delle parti

Magari! Certe volte uno si ritrova, controvoglia, da una parte ben precisa, e la sua imparzialità è solo una sottile buccia di ipocrisia.

Ma altre volte va peggio: uno si trova a condividere gli argomenti e le ragioni di entrambe le parti. Se si tratta di una rissa, in questi casi è saggio tagliare la corda prima di farsi tagliare la gola; se si tratta di una civile discussione si può sperare invece, parlando poco e con la massima prudenza, di dare il proprio contributo a risolvere la questione. Sulla carta, beninteso, perché non ho mai sentito di una discussione teorica che, da sola, abbia prodotto effetti concreti.

Sto parlando, naturalmente, della polemica che si sta sviluppando fra il Dr. Fontana, qui nella veste di fustigatore del malvezzo nazionale di impreparare i giovani per poi sbatterli allo sbaraglio nel mondo della disoccupazione muniti di un triste dimploma che pochi ormai prendono sul serio, e Piercarlo Gennero che la suddetta procedura l'ha provata sulla propria pelle uscendone giustamente furibondo. Per quel che vale, vi regalo la mia opinione: credo che vi stiate accapigliando (da galantuomini naturalmente) nel cercare di attribuirvi vicendevolmente colpe e responsabilità che non appartengono a nessuno dei due, e tanto meno ad un fantomatico "sistema", perché questo termine è diventato ormai tanto generico da non designare più nulla e nessuno. Che poi dietro a questo "sistema" si possano nascondere i veri e anonimi responsabili, magari anche in perfetta malafede, è un altro discorso, se ci sono, trovateli e metteteli alla gogna, diventerete dei pubblici benefattori.

Fabio Fumi

Egregio Signor Gennero,

fra le tante lettere concordi con quanto da me espresso in "Tribuna" su BIT n. 5, finalmente vedo la Sua, accolta dalla rubrica "Feedback", che non è dello stesso parere. Finalmente una lettera che consente, spero, di dare vita a quel dibattito costruttivo che auspicavo al termine del mio articolo in questione. Veniamo subito al sodo.

Sono d'accordo con Lei quando dice che non si possono creare buoni tecnici se essi non vengono messi in grado di esternare le proprie capacità. Ma questo, secondo me, è un preciso compito della scuola. Pare invece che Lei non sia d'accordo e pretenda che sia il mondo del lavoro ad assumersi questo onere. E arriviamo così a mettere sotto processo le aziende (piccole, medie o grandi che siano) dalla quali (uso le Sue parole) "vengono invariabilmente richieste esperienze ultrapluriennali in fantomatiche e spesso sconosciute superspecializzazioni". Questo, mi pare, vuole essere il punto della Sua critica.

Mi siano consentite, però alcune osservazioni. Lei dice che "a queste condizioni l'alta professionalità rimarrà sempre confinata ad una ristretta élite di fortunati mortali, che continueranno a palleggiarsi i posti migliori.....": che ciò avvenga, è vero. Lei ne fa cadere la responsabilità sulla "mentalità manageriale all'italiana: paura del rischio di investire su un capitale umano che potrà anche non rendere, scarsa lungimiranza, etc. etc.". A mio avviso esistono, purtroppo, entrambi i problemi, ma non così legati fra loro come Lei lascia intendere. Rimane ancora da dimostrare, infatti, che il "palleggiarsi i posti migliori" (là dove si palleggiano, beninteso), venga fatto sul piano del libero confronto fra le capacità tecniche e manegeriali dell'individuo, piuttosto che su quello dell'intralazzo: semmai si tratta di piccole o grosse camorre che prendono vita, in genere, in ambienti avvezzi alla baronie e ai giochi di potere; ma questo aspetto del problema non ci interessa e lo lasciamo a chi ha più elementi di noi per giudicare. Per quanto riguarda poi "la mentalità manageriale all'italiana", è doveroso riconoscere che in pratica il manager viene sottoposto ad un naturale processo selettivo, specie se diventa imprenditore, sempre che non si inoculi e si mimetizzi in un ambiente baronale. Il manager, infatti, è destinato a pagare in prima persona per le proprie scelte; ma oggi il manager non ha più, in Italia, il diritto di libera scelta: una volta assunta una persona, dopo quel ridicolo periodo di prova durante il quale è impossibile fare una corretta valutazione, quella persona bisogna tenersela. Questo è il vero elemento che condiziona i manager e gli imprenditori nelle loro decisioni e che, ovviamente, li fa propendere per la scelta di persone con esperienze pluriennali. Una delle principali caratteristiche del buon manager è il rispetto e l'amore per il rischio; ma il manager, come a qualunque altro individuo, si deve riconoscere il diritto al rischio calcolato. Senza questo diritto non può esistere il manager.

Lasciamo quindi perdere la "scarsa lungimiranza (del manager all'italiana) nell'individuare i potenziali cavalli di razza..... e la mancanza di considerazione per tutti coloro che hanno speso anni per il conseguimento di un obiettivo in cui credevano". Ma che cos'é questo obiettivo? Un certificato? Un pezzo di carta? O è piuttosto la capacità professionale di inserirsi in un ambiente di lavoro e dare subito le prove di tale capacità? Tutti sanno che gli studenti che escono dalla nostra scuola tecnica non hanno mai avuto per le mani un circuito integrato per vedere come funziona una gate, o il terminale di un computer per vedere se il programma, fatto al tavolino, gira o no nel computer. Lei stesso, del resto, dice "sono diplomato in elettronica ed in possesso di un attestato di *specializzazione* su microcomputers, conseguito presso un importante Istituto torinese, e nessuna esperienza nel settore". Ma che cosa significa specializzazione senza esperienza? È sicuro che questo Istituto Le abbia dato una "specializzazione" veramente degna del nome? Lei aspira ad un impiego coma analista e/o programmatore e dice di avere una discreta conoscenza del linguaggio BASIC, con alcune esperienze di programmazione. E gli altri linguaggi? Che cosa significa discreta conoscenza? Che tipo di programmatore ha fatto di Lei la scuola da cui Lei è uscita? Se è vero che Lei possiede le basi dei microcomputer, se è vero che lei conosce e ha sperimentato (a scuola) le tecniche di microprogrammazione e di interfacciamento del microcomputer, se è vero che Lei è in grado (essendo uno specialista dovrebbe esserlo) di mettere le mani su un problema pratico e di iniziarne lo studio per portarlo poi avanti passo passo fino ad una soluzione, anche approssimata, allora non

capisco come Lei, a 30 anni, faccia ancora il commesso di negozio (e neppure di stereo HI-FI).

Per concludere, caro Gennero, e lo dico con profonda convinzione, la Sua amarezza, e così quella di molti altri giovani (e non più tanto giovani) di oggi, è il diretto prodotto di una scuola superficiale e disorganizzata, dove esiste il diritto ma non il dovere allo studio. Purtroppo sui ruderi di questa scuola disastrata svolazzano come corvi istituti specializzati, centri per la formazione e l'addestramento, istituti tecnologici avanzati, etc., che approfittano di questa situazione per far scucire molti quattrini ai volenterosi, o agli ingenui, e per dar loro in cambio un pezzo di carta, nulla di più. D'altra parte non si può negare l'importante funzione, anche sociale, di molte di queste scuole (quelle serie) che danno ai giovani quel tipo di educazione che la scuola di stato non dà, o dà in modo superficiale. Ecco perché mi auguro il libero confronto fra questi istituti. "Vinca il migliore", direi in gergo sportivo. Questo libero confronto di opinioni potrebbe essere un'arma a favore della moralizzazione e maggiore professionalità della scuola tecnica, la quale è chiamata non a licenziare scontenti, ma buoni tecnici.

Giuseppe Fontana

Text processing

Sono studente universitario al mio primo anno di studi di informatica. Acquisto regolarmente BIT perché propone argomenti di divulgazione dell'informatica, nonché indicazioni di campi di lavoro interessanti per il calcolatore. Trovo inoltre utili gli articoli di hardware e, last but not least, l'introduzione delle bibliografie in alcuni articoli.

Mi piacerebbe che le bibliografie fossero sempre presenti e che sul software si ampliasse la parte descrittiva dei programmi (per esempio esponendoli anche facendo uso delle strutture di controllo utilizzate per la programmazione strutturata), permettendo così la traduzione rapida dei programmi in ogni linguaggio e l'eventuale modifica di alcune parti di essi.

Ho la possibilità di utilizzare un calcolatore per esercitazione; troverai interessante poter fare alcuni esperimenti sul text processing, ma nella biblioteca dell'università ho trovato solo studi molto avanzati nel campo del linguaggio naturale, ed il vostro interessante articolo introduttivo mancava di bibliografia. Perciò gradirei, se possibile, avere informazioni bibliografiche riguardanti l'esistenza di testi che possono essere di guida ad esperimenti di sintassi e semantica del linguaggio e che non presuppongono conoscenze ad altissimo livello.

Grazie, ed auguri per il continuo perfezionamento della rivista.

Luca Rodda
Sesto San Giovanni (MI)

Caro Luca,
riceverei le bibliografie che ti interessano e faremo del nostro meglio per rendere più esaurienti e complete quelle che compaiono su BIT.

Quanto al text processing, è un argomento pieno di fascino, inesplorato altro che marginalmente e mostruosamente vasto; temo che molto di quanto è stato scritto in proposito sia viziato da un pesante accademismo di fondo, sia cioé scritto in un linguaggio molto più involuto del necessario e terribilmente teorico; è diffcile resistere alla tentazione di inventare un'algebra specifica che magari non risolve il problema, ma appaga la vanità intellettuale dell'autore. Se troveremo delle bibliografie rispondenti alla tua richiesta, oltre a quelle relative all'articolo che citi, saremo ben lieti di fartele avere. In ogni caso questo è un APPELLO AI LETTORI: c'é senz'altro fra voi qualcuno che in proposito la sa lunga e può e vuole darci una mano ad approfondire l'argomento.
Se mi permetti di fare un pò il profeta, il text processing ti condurrà prima o poi al pattern recognition (per esaminare i concetti al di là della loro configurazione semantica) e successivamente alla traduzione automatica (quella sensata, non il vocabolario elettronico). Non che ci sia niente di male, ma sta attento: è una droga, liberarsene anche solo per un pò è peggio che tentar di smettere di fumare.

Maiuscole e minuscole

Son un giovane lettore e da poco mi interesso di microinformatica. Leggo da "sempre" BIT che è senz'altro il meglio, non aggiungo altro.

Scrivo per una curiosità che è nata leggendo la vostra rivista, ed alla quale non riesco a dare una spiegazione logica; non che sia una cosa importante, ma forse molti si sono posti lo stesso tipo di interrogativo.

In poche parole perché alcuni nomi di linguaggi relativi ai computers sono scritti per intero con lettere maiuscole, altri no?

Rinnovando i complimenti e l'invito a continuare così passo a salutare.

Claudio Bellotti - Novara

C'é una semplice regola che governa il modo di scrivere queste parole che cercherò ora di spiegare.

Quando un nome è una sigla, cioé è formato dalle iniziali di parole che compongono una frase, questo nome dovrà essere scritto tutto a lettere maiuscole; diversamente potrà essere maiuscola solo la prima lettera (se nome proprio o nome depositato), oppure tutto minuscolo.

Ecco qui il significato di alcuni dei più attuali nomi di linguaggi:

ALGOL:
ALGO*rithmic* Language
APL:
A Programming Language
BASIC:
Beginners All-purpose Symbolic In*struction Code*
COBOL:
COmmon Business Oriented Language
FORTRAN:
FOR*mula* TRAN*slator*
Pascal:
dal nome di Blaise Pascal, famoso matematico francese
PL/I:
Programming Language I
RPG:
Report Program Generator

Pare comunque che la mania di chiamare i linguaggi con le sigle stia passando di moda: una volta c'erano almeno una dozzina di linguaggi i cui nomi terminavano in OL come ANGOL e COBOL c'erano DIBOL, SNOBOL, etc. etc.).

Come vedi anche il mondo dell'informatica risente ogni tanto delle mode passeggiere.